Data reduction for the transmission of time encoded speech.

Longshaw, Stephen

Award date:
1985

Awarding institution:
University of Bath

DATA REDUCTION FOR THE TRANSMISSION OF

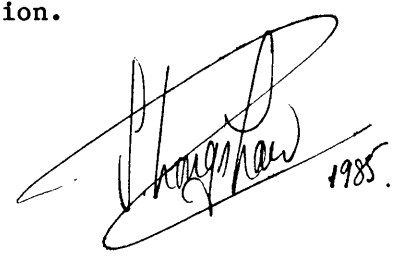
TIME ENCODED SPEECH

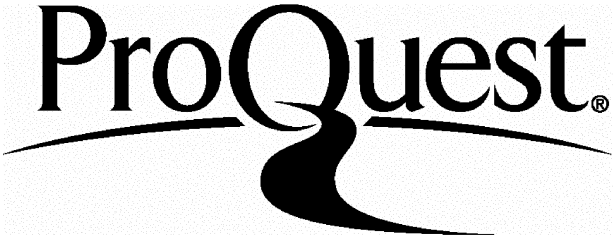
submitted by Stephen Longshaw

for the degree of Ph.D.

of the University of Bath

1985





TO

FELICITY

ABSTRACT

Time Encoded Speech (TES) transmits information concerning the duration between zero-crossings, shape and the amplitude of the signal between successive zero-crossings.

This thesis examines a number of aspects of TES with the view of achieving data reductions to enable the transmission of speech, with acceptable quality and intelligibility, at low bit rates and a practical system delay.

This thesis presents:

(i) A study of techniques for signalling amplitude information in a TES coder. It was indicated that a minimum of the order of 1 bit per epoch is required. Diagnostic Rhyme Tests (DRT) yielded intelligibility scores of the order of 88% for algorithms employing 1 and 2 bits of amplitude information per epoch.

(ii) Investigations into Median and Moving Average filtering for pre-processing the epoch duration sequences. It has been shown that such applications, which involve simple numerical smoothing, are of little value for they degrade the quality of the synthesised speech.

(iii) Studies of Extremal Coding and Orthogonal Transformations for achieving data reductions in the signalling of epoch duration and, in some instances, the peak magnitude sequences. Each technique yielded a useful data reduction. The technique using

Hadamard Transformations yielded the greatest data reduction, a ratio of 2:1 for the representation of the epoch duration sequences. The Hadamard Transformation also proved to be of low complexity in its implementation.

(iv) A real-time simplex digital voice channel, developed during the course of this thesis, and a study of the implementation of TES and TES related coders. It is reported that speech of acceptable quality and intelligibility is achieved for a transmission rate of 10 or 15kb/s with a transmission delay of 300ms.

LIST OF CONTENTS

List of Symbols and Abbreviations

x(n)	: Discrete sample at time instant n.
x'(n)	: Processed x(n).
A_I	: Amplitude parameter of an epoch.
A'_I	: Processed A_I.
A_N	: Amplitude parameter for group of epochs.
I	: Transmitted amplitude information per epoch.
CAPC	: Chance Adjusted Percent Correct.
.LT.	: Less than.
.LE.	: Less than or equal to.
.EQ.	: Equal to.
.GE.	: Greater than or equal to.
.GT.	: Greater than.
Tx.	: Transmitter.
Rx.	: Receiver
ADC	: Analogue-to-Digital Converter.
DAC	: Digital-to-Analogue Converter.
DRT	: Diagnostic Rhyme Test.
DCT	: Direct Comparison Test.
k	: Kilo, or 1,000.
ms	: Millisecond, or 0.001 seconds.
μs	: Microsecond, or 0.000001 seconds.
ns	: Nanosecond, or 0.000000001 seconds.
Hz	: Hertz.
dB	: Decibel.
FWT	: Fast Walsh Transform.

FHT	: Fast Hadamard Transform.
X(N)	: Column Input Vector.
Y(N)	: Column Transform Vector.
y_n	: nth coefficient of a transform vector.
CSF	: Coded Speech File.
ADalphabet	: Al-Doubooni coding alphabet.
KHalphabet	: King and Holbeche coding alphabet.

Chapter 1

Introduction and Review of Speech Communication Systems.

1.1 Introduction

Speech is a communication technique unique to man. It is our main method of conveying thoughts, ideas, concepts and facts to other humans. Of the many thousand species of life sharing this world only man has developed the vocal means of communication beyond the rudimentary stage.

In personal, face-to-face conversation, we have no need to process speech with sophisticated equipment. However, the concept of speaking over long distances has intrigued man for generations.

In our early history, communication between two people who were some distance apart was achieved by sending messengers with verbal information, eliminating the need for the communicators themselves to travel. The messenger had to remember the verbal information until it could be delivered. With the advent of writing, the necessary information could be inscribed on paper, wood, or stone and "mailed". In this case, the storage medium was probably more reliable. However, as in the first example, the communication method (ie. by messenger) was such that long time delays were still inherent in the transmission of data.

The direct, "real-time" conversational method of exchanging information over distances was still a highly desirable mode of communications for man, as it would allow an immediate feedback of a response enabling information to be clarified, and hence understood.

Numerous techniques ranging from smoke signals to semaphore

were devised for communicating over long distances. Yet despite the desire and motivation to accomplish this, it was not until man learned to generate, control and convey electric current that "real-time" communications became possible with the advent of telephony and its associated communication link, or "channel".

Initially, audio amplifier, two wires, and appropriate relays to complete the connections for transmitting the electrical analogue of the speech was adequate. However, as the distances between speakers increased, more sophisticated equipment was necessary and the transmission medium became a radiated carrier wave, modulated by the speech waveform. With this system the information might now be subjected to corruption by noise bursts over the link. However, the time delay in transmission was sufficiently short for the errors to be corrected by the simple technique of re-transmission.

Economic developments and social demands led to the growth of a host of different channel media, each promising increased capacity, with better quality of communications over long distances. Today, society has accepted the role of communication channels as common place for long distance communications, although the problems of time delay were evident during communications between NASA's Mission Control and the Apollo astronauts on the moon.

In todays society communication systems, in particular voice communications, are at a premium. The emergency services, armed forces, security organisations and many other businesses demand reliable portable two-way communications. More recently, with the legalisation of Citizen Band (CB) radio, portable two-way communica-

tions have been made available to a host of professions and the general public whose requirements could not be adequately catered for by a telephone link.

Demands on available channels are very high and continue to increase, and society mistakenly assumes that communication channels are unlimited, but they are not. The limitations are either space, or time, or the economics of supplying thousands of channels. Some communication links, such as a satellite link, are by their very nature expensive.

Communication Engineers strive to ensure that the optimum use is made of the available communication channels, and look for means of making such communications more effective. One way to do this is to ensure that only useful information is transmitted, and that redundant information is eliminated. Because the dominant communications traffic today is speech, engineers and scientists are continually researching techniques of speech and data compression and/or redundancy reduction by speech coding to achieve a reduction in data storage and/or signal bandwidth requirements, thereby allowing more conversations to be stored or carried over the same channel.

Speech signals are composed of sounds. These sounds and the transitions between them serve as a symbolic representation of information. The rules of language govern the arrangement of these sounds (symbols). Linguistics is the study of these rules and phonetics is the study and classification of the sounds of speech. Although a study of linguistics and phonetics is very important especially in the areas of speech recognition such a study would take us too far

afield. However, in processing speech signals to extract or enhance information, it is helpful to have as much knowledge as possible about the structure of the signal. Thus it is instructive to review the nature of speech production and the main classes of speech sound before we progress to survey the more "popular" techniques employed for speech coding.

1.2 Speech Production[1-3]

Making a sound is not necessarily making speech, no one can talk through a flute. It is necessary to articulate, to join together the basic note or notes before they are transformed into speech. The so-called "mobile articulators" are the lower lips plus its jaw, the various parts of the tongue, and the vocal cords. The mobile articulators have to move towards something - the "fixed articulators". These are the upper lips, upper teeth, the ridge of gum supporting the teeth, the hard palate at the front and the soft palate at the back of the mouth and the sides of the larynx. A cross-sectional view of the vocal mechanism showing some of the major anatomical structures involved in speech production are shown in figure 1.1.

Try to say 'pop' without the lips touching, or 'that' without pressing the tongue's tip to the upper teeth, articulation is inevitable. The articulators form the vocal tract which is an acoustical tube of non-uniforn cross-sectional area. It is terminated at one end by the vocal cords and at the other end by the lips. An ancillary 'tube', the nasal cavity, can be connected or disconnected by the movement of the soft palate. The basic source of acoustic power

is the lungs. During exhalation they force out a stream of air via the trachea through the vocal mechanism to form one of three sounds, each of which are described in the following sections.

1.2.1 Voiced Sounds

In a relaxed condition, the vocal cords move apart leaving a large opening, while during speech they draw together to form a split-like orifice called the glottis. In voiced sounds, the vocal cords are initially tensed and together. The lungs force air up the trachea, thus increasing the pressure behind the vocal cords (this is known as the sub-glottal pressure) until it is sufficient to force the vocal cords apart resulting in lateral acceleration of the air through the narrow glottis. On entering the wider pharynx, the air slows down. The velocity differential brings about a reduction of pressure in the glottis. The reduced pressure enables the vocal cords to return to their initial position, and the air flow diminishes. The sub-glottal pressure then begins to increase again and the whole cycle is repeated.

The vocal cord mass, tension and sub-glottal pressure essentially determine the period of vibration. This repetative action produces quasi-periodic pulses of air which causes excitation of the vocal tract and generates the voiced sound of speech. The vocal tract is similar to a resonant cavity and therefore intensifies the energy at certain bands of frequencies. These resonances, whose frequencies are altered by movement of the articulators are called the "Formant frequencies" or simply " Formants". Voiced sounds are

usually characterised by three or four formants in the frequency range up to approximately 4.0kHz. An example of the periodic nature of a voiced sound is given in figure 1.2.

1.2.2 Fricative Sounds

Fricative sounds are generated by forming a constriction at some point in the vocal tract and forcing air through the constriction at a high enough velocity to produce turbulance. This creates a broad spectrum noise source to excite the vocal tract. The sound produced is different from the voiced sound as its method of production is inherently random and does not exhibit any repetative structure.

The constriction in the vocal tract may be formed between the teeth and lips as in 'f' or between the tongue and hard palate as in 'sh'. The position of the constriction has a profound effect on the characteristic sound radiated from the mouth.

If the noise source alone is used to generate the fricative, the fricative is unvoiced and exhibits a 'breathy' and 'hissy' quality. If the vocal cord source operates in conjunction with the noise source, the fricative is voiced. An example of unvoiced frication is given in figure 1.3.

1.2.3 Plosive Sounds

The third sound source are the plosives or stop consonants.

These sounds are produced by stopping the air flow from the lungs by completely closing off the vocal tract, building up pressure behind the closure and quickly releasing it thereby causing a transient excitation of the vocal tract which results in the sudden production of sound. During the build up of pressure, if the cords are vibrating, then the plosive is preceded by low level sound. If the cords are not vibrating during the pressure build up the plosive is preceded by silence. These two type of plosives are termed voiced and unvoiced, respectively. Table 1.1 presents a classification of the various speech sounds.

1.3 Speech Bandwidth Compression

This section defines channel capacity and indicate its relevance when discussing speech bandwidth compression. The catagories under which speech coding techniques are classified will be described before reviewing some of the most important of the currently known techniques.

Speech sounds occupy the frequency range from approximately 50Hz to 10kHz. A bandwidth of 10kHz is therefore required for transmission of the exact electrical analogue of a speech wave. In order to utilise the full capacity of todays communications channels where bandwidth capabilities of WHz, W is much much greater than 10kHz, are common. Multiplexing [6] of conversational speech is employed. However, the channel capacity soon becomes exhausted when employing such simple techniques. To further increase the number of conversations transmitted over the channel, the speech band capacity occupied

by each conversation must be reduced. This has been, and still is the goal of many researchers in the broad 'field' of speech communications.

Channel capacity is measured in binary units of information (bits) per second , b/s, and specifies a maximum rate at which the bits can be transmitted over a channel with errors reduced to negligable proportions. Shannon[7] has shown that the theoretical maximum capacity, C, of a communications channel of bandwidth, W, for a signal-to-noise power ratio, S/N, to be:

$$C = W.\mathrm{Log}_2(1 + S/N) \qquad b/s \qquad (1.1)$$

For a given signal-to-noise ratio the theoretical maximum channel capacity and bandwidth are directly related. Techniques for the reduction of required channel capacity are often termed 'Bandwidth Compression' or 'Bandwidth Reduction'. The achieved bandwidth reduction is, in general, specified as a ratio of the previous channel capacity and the currect channel capacity.

A question often posed when discussing the achieved bandwidth compression is, "How significant is this result?". The answer to this depends upon the envisaged area of application. For example, consider a speech system which yields good quality speech with a transmission rate of 16kb/s. In a commercial environment a significant bandwidth compression is reducing the transmission rate from 16 to 8kb/s, a ration of 2:1, and yield speech of the same quality. However, for military applications fair to good quality speech at a transmission rate of 12kb/s, a ratio of 1.33:1, is highly significant

because the 4kb/s 'save' may be utilised for conveying secure data.

The initial techniques proposed for reducing the required channel capacity employed analogue processing and the reductions were achieved by exploiting known characteristics of conversational speech. However, for digital processing greater reductions were achieved using Analysis-Synthesis techniques, a form of source coding. Source coders in this application make no attempt to preserve the original waveform, instead the input speech is analysed using a priori knowledge of how the signal is generated at the source. The speech parameters are transmitted and, at the receiver, employed to synthesise the speech signal. Source coders of this nature for speech are generally referred to as a Vocoder, a term derived from the words VOice CODER.

Reductions approaching those of source coders can be attained using waveform coding. As the name implies, the coder essentially strives to copy the actual shape of the waveform. In principle they are designed to be signal/source-independent and therefore can code a variety of signals equally well eg. speech, music, electrocardiograms. Waveform coders tend to be robust for a wide range of speaker characteristics and for noisy environments.

There are many ways of combining some of the detailed signal description possibilities of waveform coders with some of the redundancy exploitation of source coders. The resultant systems, known as Intermediate coders, generally yield comparable speech quality at data rates intermediate to that of vocoders and waveform coders.

The following sections briefly review some of the most important of the commonly adopted methods of encoding speech signals. Sections 1.3.1 and 1.3.2 review source and waveform coders. Section 1.3.3 describes an intermediate coder known as Time Encoded Speech; a relatively new digital speech encoding technique.

1.3.1 Source Coders, Vocoder[1,8-12]

Systems which attempt to preserve the short-term power spectrum of speech while disregarding most of the phase information, are called Vocoders. Vocoders strive to retain the perceptually significant properties of the waveform with the intention of synthesising a signal at the receiver which sounds like the original.

The traditional model of speech production, figure 1.4, is one where the source of the sound and the resonant system which modifies the sound are separable and do not interact.

For vocoder purposes, speech sounds are classified as either voiced or unvoiced. The voiced sounds are represented by a periodic pulse generator and the unvoiced sounds by a random noise generator. These sources are normally considered mutually exclusive with a signal to indicate switching between the voiced and unvoiced sources. The intensity of sound excitation of each source is represented by an amplitude or gain signal. In addition, the periodicity or 'pitch' of the voiced pulse must be signalled.

The important difference, in principle, between vocoders lie in their representation of the vocal tract transfer function. This

description can take a variety of forms. In the Channel vocoder, values of the short-term amplitude spectrum of the speech signal are evaluated at specific frequencies. The Linear Predictive Coding (LPC) vocoder utilises linear prediction coefficients which describe the spectral envelope, whilst the Formant vocoder uses frequency and amplitude values of major spectral resonances. Autocorrelation vocoders specify samples of the short-term autocorrelation function whilst coefficients of a set of orthogonal functions that approximate the speech waveform comprise an Orthogonal Function vocoder. There are numerous other variants. With all these methods of waveform description the data is coded into 'frames' associated with the spectra measured at intervals of 10 to 30ms.

A basic channel vocoder is shown in figure 1.5. A series of bandpass filters are used to divide the speech signal into frequency channels. The number of filters employed within the analyser or synthesiser vary depending upon the application. The first channel vocoder demonstrated by Dudley[8] had only 10 equal bandwidth spectrum channels covering the speech band upto 3kHz, while the JSRU channel vocoder[11] had 19 non-uniformly spaced channels which had a rough correspondence with the critical bands of auditory perception in the range 0.25 to 4.0kHz. The signal components in each of the channels are full wave recified and low pass filtered to yield a continuous estimate of the speech power spectrum amplitude in each channel. Independent to the spectral analysis, the fundamental frequency is measured and a voiced/unvoiced condition determined by the voicing detector.

At the receiver (or synthesiser) the speech is synthesised using estimates of the power spectrum together with the pitch and voicing information. During voiced segments pulses at the pitch rate are output from a pulse generator to excite a bank of filters. The excitation is adjusted using the power spectrum amplitude in an attempt to equalise the output energy to that measured for the corresponding channel in the analyser. For unvoiced sounds, a gaussian noise source excites the filter bank. The output of each filter is then combined to produce artificial speech similar to the original. An overall bandwidth compression of the order of 10:1 in the transmitted signal is possible.

As digital technology evolved and small processors became more economical, powerful and faster so their applications in digital signal processing increased. Most modern vocoders employ a digital transmission path. The Channel and LPC vocoders can produce useable albeit poor quality speech at transmission rates of 2.4kb/s. It is generally recognised that good speech quality can be achieved at data rates as low as 1.2kb/s, by Formant vocoders. However, the main difficulty with Formant vocoders lie in the automatic formant analysis during consonant or vowel/consonant boundaries where discrepancies in signalling can arise [12].

Vocoders tend to be fragile (in terms of parameters such as voiced/unvoiced decisions and pitch values), the performance is often talker- and environment-dependent and the output speech quality, generally, has a synthetic (less than natural) quality. These characteristics and the signal model utilised constitute a ceiling on the

performance that vocoders can achieve.

1.3.2 Waveform Coders

Waveform coders represent an analogue waveform in digital form for transmission. This digital representation offers ruggedness, efficient signal representation, ease of encryption, the ability to combine transmission and switching functions and the advantage of a uniform format for different types of signals.

(i) Pulse Code Modulation, PCM[13,14]

Historically, PCM was the first method used to digitally represent speech waveforms and is still widely used. The sampling theorem states that a signal whose highest frequency of importance is f_h can be completely specified by $2f_h$ samples taken at equal intervals of $1/2f_h$ seconds. PCM invokes the sampling theorem by uniformly sampling the (bandlimited) waveform at a minimum rate of at least twice the highest frequency in the waveform. The amplitude of the sample is quantised into one of 2^B levels. These discrete amplitude levels are represented by unique binary words of length B bits.

To decode the PCM signal, the binary words are mapped back into amplitude levels creating an amplitude pulse sequence. This sequence is low pass filtered with a filter whose cut-off frequency is equal to the highest frequency in the input waveform.

PCM is illustrated in figure 1.6. The difference between the

original and quantised signal is regarded as noise and termed 'quantisation noise'. It has been shown [14] that the mean square quantisation noise, Q_n, is a function of step size, S, where :

$$Q_n = S^2/12 \qquad (1.2)$$

If the 2B amplitude levels are uniformly spaced, low level signals have a poor signal-to-quantisation noise ratio compared with large amplitude signals. In order to combat this problem a number of techniques have been developed, namely Non-Uniform Quantisation, Adaptive Quantisation and Instantaneous Companding.

Non-uniform quantisers are characterised by fine quantisation steps for the frequently occurring low amplitudes in speech; while much coarser quantisation is used for the less frequent large amplitude excursions in the speech waveform.

Adaptive quantisation utilises a quantiser whose characteristics (uniform or non-uniform) shrink or expand in time like an accordion. In an adaptive quantiser with a one word memory [15] the adaption of step size s(n+1) is characterised by:

$$s(n+1) = s(n).M(|\text{Quantiser Output}(n)|) \qquad (1.3)$$

Where M is a function only of the latest quantiser output.

The use of a non-uniform quantiser is equivalent to the presentation of a compressed signal to a uniform quantiser and a subsequent expansion of the output. Smith [16] investigated instantaneous companders and demonstrated that a mu-law (-law) compression characteristics using 128 quantisation levels (7 bits per sample) could

provide "toll quality speech" of similar performance to that of a 2048 level (11 bits per sample) quantiser. Figure 1.7 shows the input-output relation for mu-law characteristics.

Civil telephony systems employ 8 bits per sample quantisation with a sampling rate of 8 kHz, resulting in a transmission rate of 64 kb/s.

(ii) Differential PCM,(DPCM)[1,17-20]

Differential Pulse Code Modulation (DPCM) estimates the next sample based on the previously transmitted samples. The estimate is subtracted from the actual sample value to give a prediction error. The prediction error is then quantised, coded and transmitted. At the receiver, the original signal is synthesised from the sequence of quantised prediction errors. A block diagram of a DPCM system is shown in figure 1.8.

The predictor weights past sample values so as to minimise the average energy of the prediction error or difference signal. The 'weights' are calculated using long term statistics of a representative sample of speech. Once selected, the 'weights' remain fixed for any given DPCM system.

The application of adaptive quantisation in DPCM systems has been investigated [20] and the quality of speech output from a 4 bit adaptive differential PCM (ADPCM) coder was subjectively judged to be better than that of 6 bit log-PCM.

(iii) Adaptive Predictive Coding,(APC)[1,21,22]

Adaptive Predictive Coding (APC) differs from ADPCM in that, besides the quantiser being adaptive, the predictor in APC is also adaptive and is updated periodically. Atal and Schroeder [21] employed an adaptive linear predictor which was updated every five milliseconds to minimise the mean square error between the predicted and the true values of the signal. The predictor parameters were transmitted periodically along with the difference signal and quantiser step size.

The block diagram of an APC system is identical to that of figure 1.8 when, the predictor in figure 1.8 is replaced with that of figure 1.9. This configuration was first postulated by Atal and Schroeder based on the observation that speech signals contain both long term and short term redundancies due to excitation or pitch signal and the vocal tract, respectively.

Subjective comparisons with speech from a 6 bit log-PCM encoder [21] indicate that the quality produced by APC is very similar. More recently Atal [22] has shown that predictive coders have the potential of producing superior performance to that of source coders, even at low bit rates.

(iv) Delta Modulation,(DM)[1,23-25]

Delta Modulation (DM) is a predictive coding system similar to DPCM. However, in DM the signal is sampled at many times its Nyquist

rate and each sample is encoded employing one bit (2 levels). A DM system generates a local copy of the input waveform and successsively modifies this copy, as specified by a digit code, in an attempt to track the input. A block diagram of a DM system is given in figure 1.10. The step size, SS, is fixed for non-adaptive (linear) delta modulation (LDM).

Figures 1.11(a),(b) and (c) are waveforms produced at various points within the DM system of figure 1.10. Figure 1.11(a) compares the local copy of the input waveform (which is the output of the integrator, s(t)) with the input waveform itself. Figure 1.11(b) is the output of the sampler, e(n), while figure 1.11(c) gives the error signal derived by differencing the input signal, x(t), and the local copy, s(t).

As shown in figure 1.11(c) the quantisation noise takes two distinct forms. The first, 'granular noise', occurs when the DM coder is tracking a constant or slowly varying signal. The second, 'slope overload noise', occurs when the slope of the input signal is greater than the maximum slope of the DM system. The optimum step size is one which achieves an appropriate balance between slope overload which is dominant when the step size, SS, is small and granular noise when SS is large.

To reduce these noise sources a number of techniques for altering the step size, generally refered to as Adaptive Deltamodulation (ADM), have been investigated [24]. The adaption can be either continuous as in Continuously Variable Slope Delta modulation (CVSD) which varies SS over a continuous range, or discrete as in Variable

Slope Delta modulation (VSD) which adapts, depending the error signal, by changing SS in steps of 2^n.

The data rates necessary for DM to produce high quality speech are relatively high. For low data rates, adaptive delta modulators can be designed to have a better signal to noise ratio than that of 7 bit log-PCM. However, ADM typically requires 16-20kb/s for satisfactory speech production.

1.3.3 Intermediate Systems

In the last decade many of the speech encoding systems developed were of the Intermediate type[26-29]. Of these some of the better known are still being extensively researched. Intermediate coders combine some of the detailed signal description possibilities of waveform coders with some of the signal redundancy exploitations of vocoders in an attempt to achieve a system of low complexity which has a high tolerance to noise and low data transmission rate. A relatively new technique for digitally encoding speech which may be classified as an intermediate coder is described below.

Time Encoded Speech,(TES)[30,31]

In 1948 Licklider and Pollack published the results of their investigations into the effects of "infinite peak clipping" on the intelligibility of a speech waveform [32,33]. It was reported that a speech waveform, clipped such that no amplitude information remained, yielded 75% random word intelligibility. If differentiated prior

to clipping the intelligibility was increased to 97%. The only information transmitted was the position of the zero-crossings of the speech waveform. At the receiver, this information was decoded and used to generate "square speech", that is rectangular waves of constant amplitude, alternating in polarity. However, this technique has been of little practical use for two main reasons. The waveform's zero-crossings must be located accurately to maintain intelligibility and an acoustic noise background caused severe degradation of the signal.

In 1978, King and Gosling described a new method of encoding speech waveforms called Time Encoded Speech (TES). This technique was akin to rectangular or "square" speech in that the speech signal was segmented between successive real zeros of the function. However, for each segment of the waveform, the code consisted of a single digital word derived from two parameters of the segment: its quantised time duration and its shape.

Measurement of the interval between real zeros of the waveform was straight forward. This parameter is often referred to as the 'epoch duration' or 'epoch length' and its units are either the number of samples between real zeros or the physical time between real zeros (ie. the product of the number of samples between real zeros and the sampling interval). The epoch durations were logarithmically quantised so that, in absolute terms, the shorter epoch durations were transmitted more accurately than the longer, thus maintaining an approximately constant fractional accuracy.

TES differed from the non-linearly quantised rectangular speech

in that, in addition to the quantised epoch durations, epoch shape descriptors were also transmitted. This provided additional waveform information and extra psycho-acoustic cues to enhance the naturalness of the synthesised speech.

In principle, the epoch shape could be compared with a catalogue of shapes and a code selected which represents the shape in the catalogue nearest to the actual segment shape. However, since the short-term phase relationships, which greatly affect waveshape, are hardly perceived at all by the ear, epoch shapes which look different when viewed on an oscilloscope may not sound different when heard. As a result of this, much simplification of shape identification is permissible. The product of the number of durations multiplied by the number of distinguishable epoch shapes constituted a large 'alphabet' of distinguishable symbols. Thus such a simple coding strategy would have been of limited value. However, further research revealed that the large number of naturally occurring symbols could be mapped to a smaller number. A two dimensional matrix, corresponding to quantisation of time on one axis and a list of shape descriptors on the other, permitted mapping onto a reduced alphabet of symbols distributed uniformly over the matrix. The reduced alphabet approximated the natural symbols which are mapped onto the matrix.

Synthesis was a relatively simple matter. Stored epoch shapes were reproduced in sequence at the correct duration, in accordance with the shape and duration specifications of the received codeword. This yielded waveforms of fixed amplitude.

Mean amplitude is a relatively slowly varying characteristic

of speech waveforms. To further improve the naturalness of the synthesised speech, after every eighth transmitted codeword an additional codeword was inserted which communicated the mean speech amplitude of the next eight codewords and the synthesis mean amplitude was reset to this new value.

It was reported that for a reduced alphabet of 23 codewords simulation of encoding and synthesis (utilising a PDP 8/e minicomputer) yielded speech of "good intelligibility and speaker identifiable" at an estimated bit rate of 5.4kb/s.

Time Encoded Speech involves the coding of variable-rate source signals into constant-rate signals for transmission. Buffer storage at transmitter and/or receiver is required to permit this "variable-rate to constant-rate" transformation. As a result, delays are introduced into the communication process.

The mechanisms involved, the influences of buffer size and the delays associated with such systems were described by Turner et al [34], who concluded that "fairly long delays (of the order of one second) may be involved in the distortionless transmission of speech encoded using information about the waveform segments linkng successive real zeros from speech signals". Mason and Balston [35] extended this analysis, proposing that a modest increase (by a factor of 1.5 to 1.9) in transmission rate over the average source generation rate would permit the distortionless transmission of time encoded speech with a tolerable system delay (200ms). Turner et al [36] derived explicit expressions for the limiting delay in time encoded speech type systems operating at data transmission rates higher than the

average rate at which the information is produced by the variable-rate source. However, King and Holbeche [37] offered experimental evidence which indicated that transmission delays in time encoded speech may be reduced an order of magnitude by utilising suboptimum bounded entropic codes in place of constant length codes.

Despite the initial promise of Time Encoded Speech the results of simulation exercises were dissappointing and many significant questions were left unanswered, for which there was no prior art to turn to for guide-lines.

Time Encoded Speech offers a method of medium complexity for the transmission of digitally encoded speech. Table 1.2 shows the relative advantages and disadvantages of a number of speech coding systems, a number of which were omitted in this review, and highlights the need for low to medium complexity intermediate coders.

1.4 Thesis Organisation

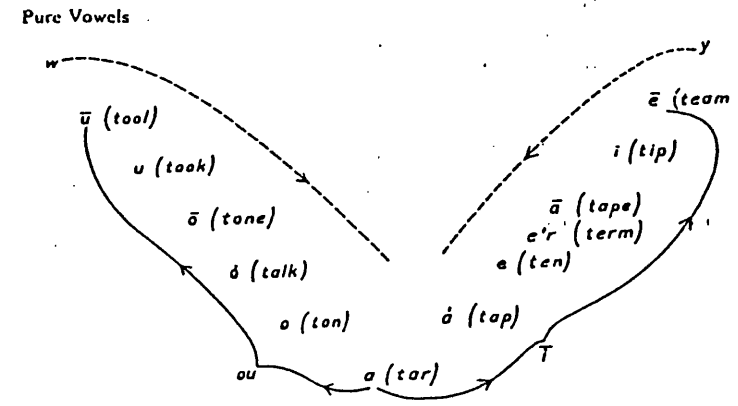
This thesis details an investigation into data compression techniques for the reduction of data transmission rate or buffer-delay associated with Time Encoded Speech. Chapter 1 has highlighted the need for bandwidth compression, given a brief 'tutorial' on speech production and finally reviewed some of the better known speech coding techniques to provide a benchmark against which TES may be evaluated.

Chapter 2 describes the simulation algorithms and modification of the coded speech data structures implemented for these investiga-

tions. Details of a microcomputer system developed in parallel to and utilised within the investigations reported here for the real-time evaluation of Time Encoded Speech are also presented.

Chapter 3 describes the real-time investigations into the amplitude information requirements of TES. Chapters 4, 5 and 6 describe simulation investigations into possible data reduction techniques applied to the sequences of epoch parameters, where: Chapter 4 details the study of Median and Moving Average Filters for the preprocessing of the epoch duration sequences for the enhancement of the effectiveness of data compression techniques; Chapter 5 presents the investigations into Extremal Coding of the sequences of epoch durations and peak magnitude parameters; Chapter 6 presents the study of Orthogonal Transformations of the epoch durations sequence and inspected possible bandwidth reduction techniques.

Chapter 7 presents the real-time implementation of the Al-Doubooni and King and Gosling TES coders, and the TES related coders which have incorporated those techniques of significance to emerge from chapters 3 to 6. Chapter 8 presents the conclusions and recommendations for areas of further research.



Combinational and Transitional Vowels
w — y — ou — i — h

Semi-vowels
l — r

Stop Consonants

Voiced	Unvoiced	Nasalized	Formation of Stop
b	p	m	lip against lip
d	t	n	tongue against teeth
j	ch	—	tongue against hard palate
g	k	ng	tongue against soft palate

Fricative Consonants

Voiced	Unvoiced	Formation of Air Outlet
v	f	lip to teeth
z	s	teeth to teeth
th (then)	th (thin)	tongue to teeth
zh (azure)	sh	tongue to hard palate

Table 1.1 : Classification of Speech Sounds (After Crandall [5]).

CLASS	EXAMPLES	BIT RATE kbit/s	QUALITY	COM-PLEXITY	CURRENT STATUS
Waveform	PCM	64	excellent	low	widespread use
	adaptive delta modulation	32	very good		
	DPCM	16	fairly good		
	(with companding or level adaptation)	8	poor		demo. available
Analysis/ synthesis	Channel vocoder	2.4	fair	high	widespread use
	LPC vocoder	2.4	fair	high	some use, increasing
	Formant vocoder	12	can be good	very high	research
	Pattern-matching vocoder	0.8	poor so far	very high	research
	Phonetic vocoder	0.1 - 0.2	poor so far	very high	research
Intermediate	Baseband vocoder	9.6	good	fairly high	demo. available
	APC	16	very good	fairly high or high	designs available - research continues
	or	8	good		
	RELP	4.8	fairly good		
	Sub-band coding	16	good	medium	research
	SBC with harmonic compression	8	good	high	research
	Adaptive transform coding	16	very good	high	research
		8	good		
		4.8	fairly good		
	ATC with harmonic compression	8	very good	high	research
		4.8	good		
	Time-encoded speech	4-16	not yet proved	medium	research

Table 1.2 : Summary of the Properties of Speech Coding Systems (After Holmes [38]).

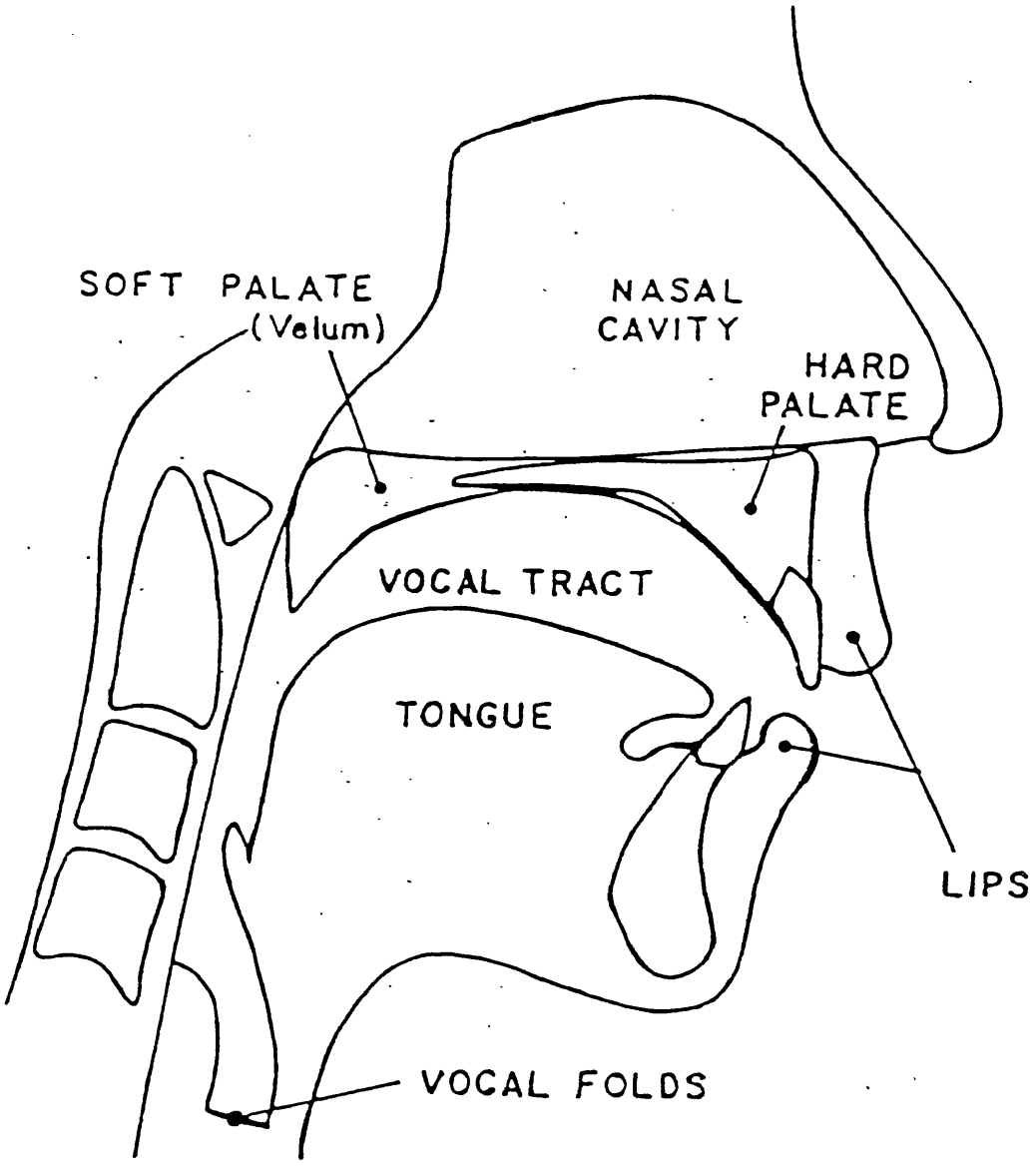


Figure 1.1 : Cross-sectional view of the vocal mechanism showing some of the major anatomical structures involved in speech production (After Markel and Gray [4]).

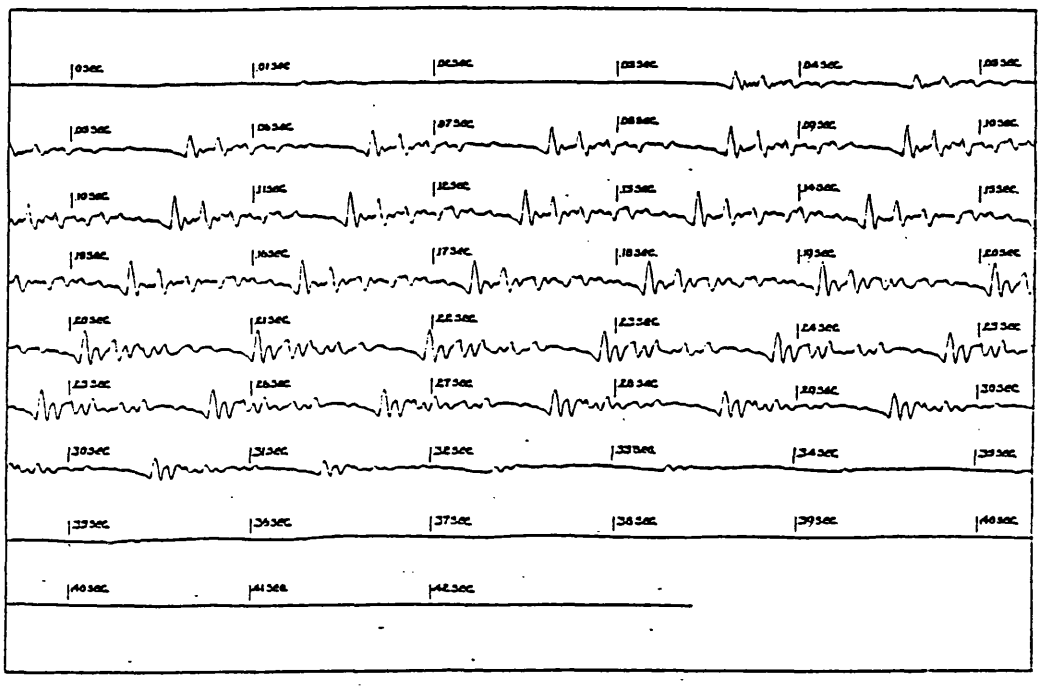


Figure 1.2 : "ar" as in part. Low pitched male speaker.

(After Crandall [5]).

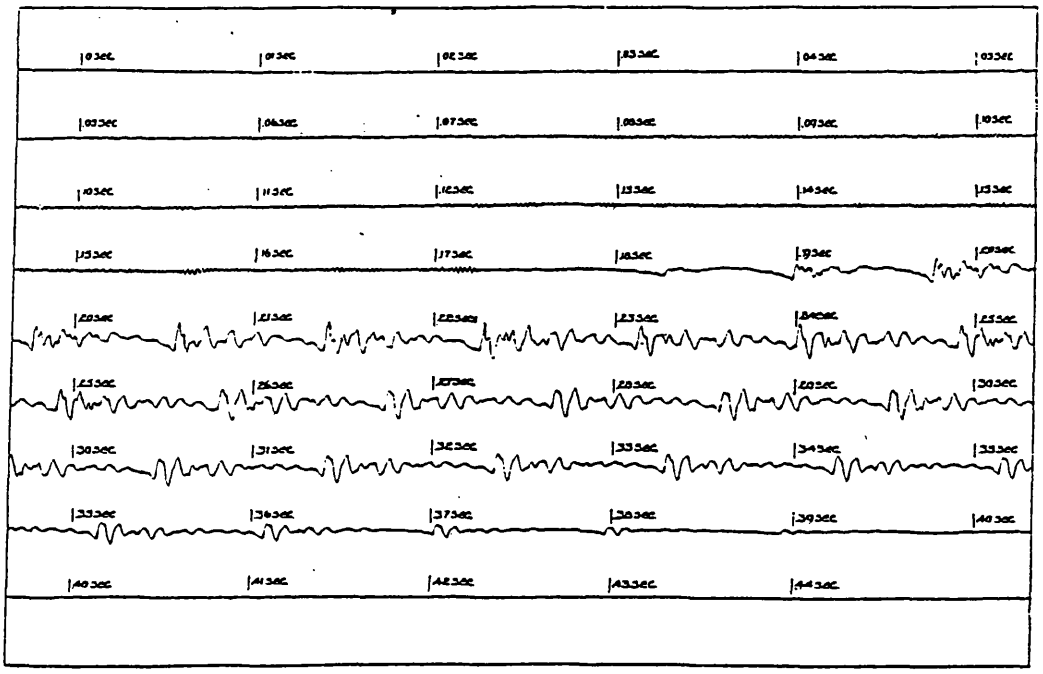


Figure 1.3 : "Sa" as in sat. Male speaker.

(After Crandall [5]).

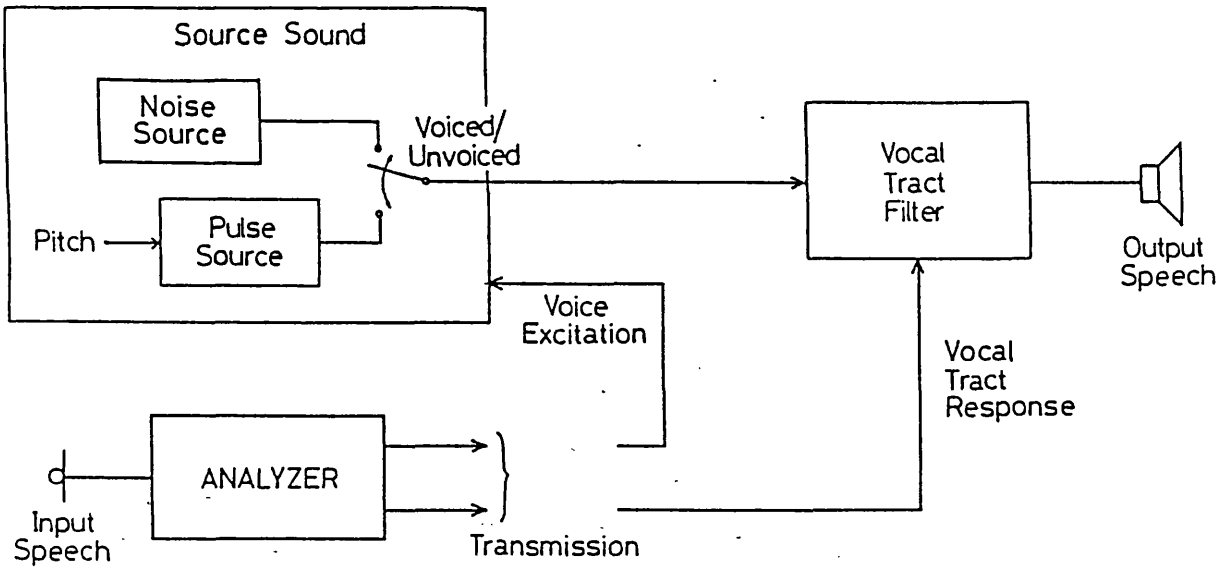


Figure 1.4 : Source-System Representation of Speech Production

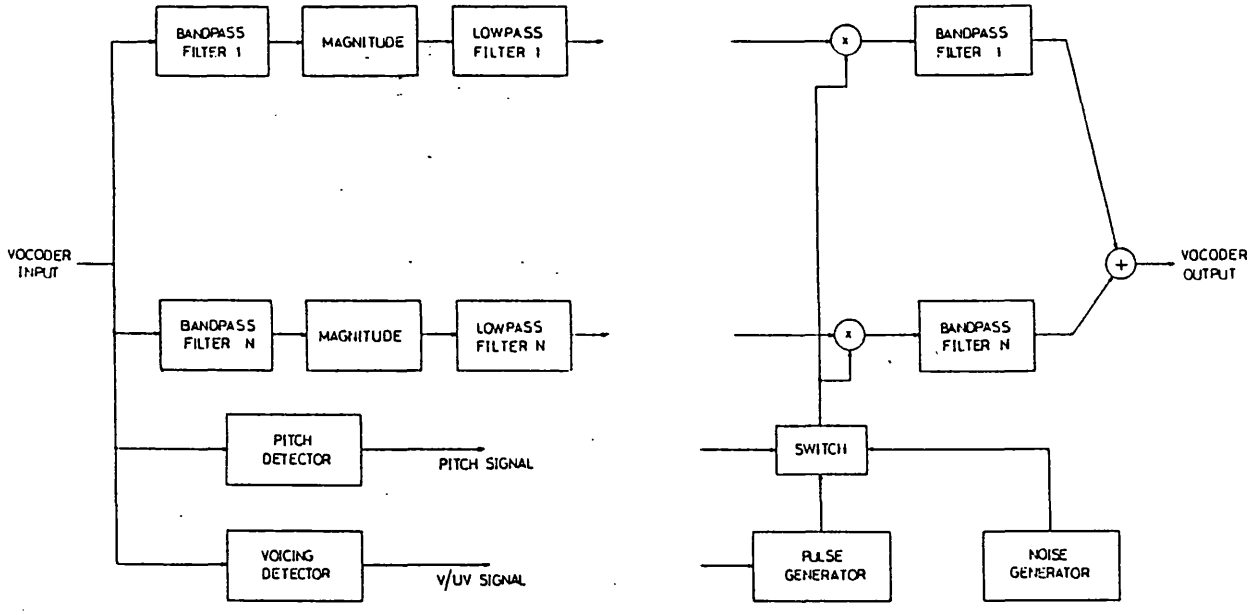


Figure 1.5 : Block Diagram of a Channel Vocoder.

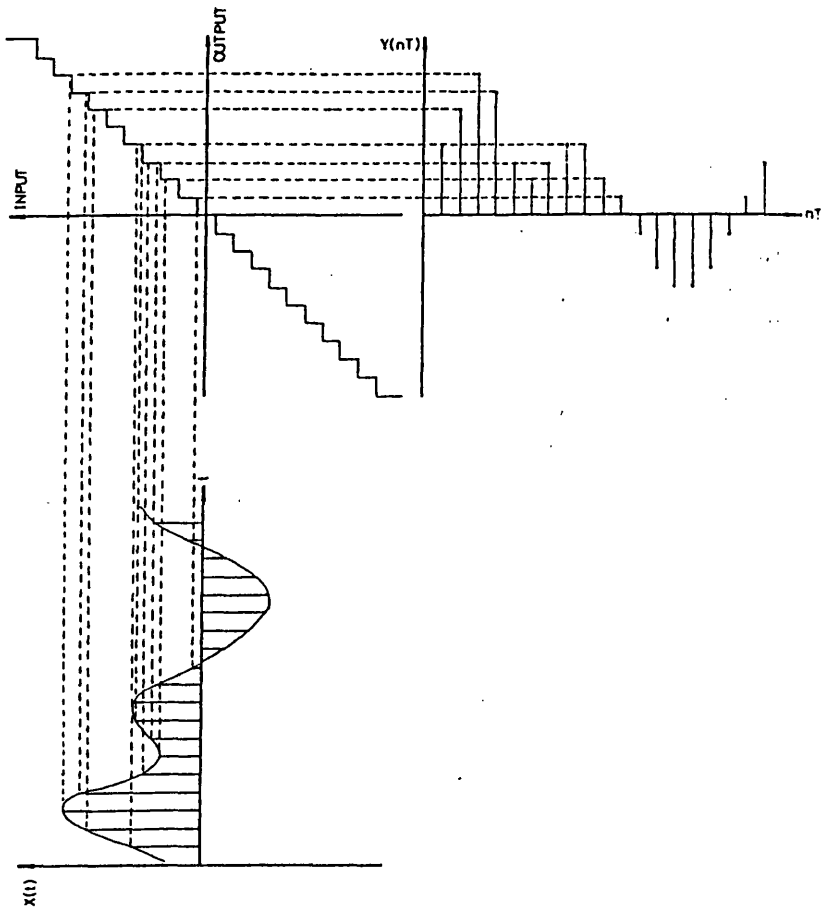


Figure 1.6 : Illustration of Pulse Code Modulation (PCM).

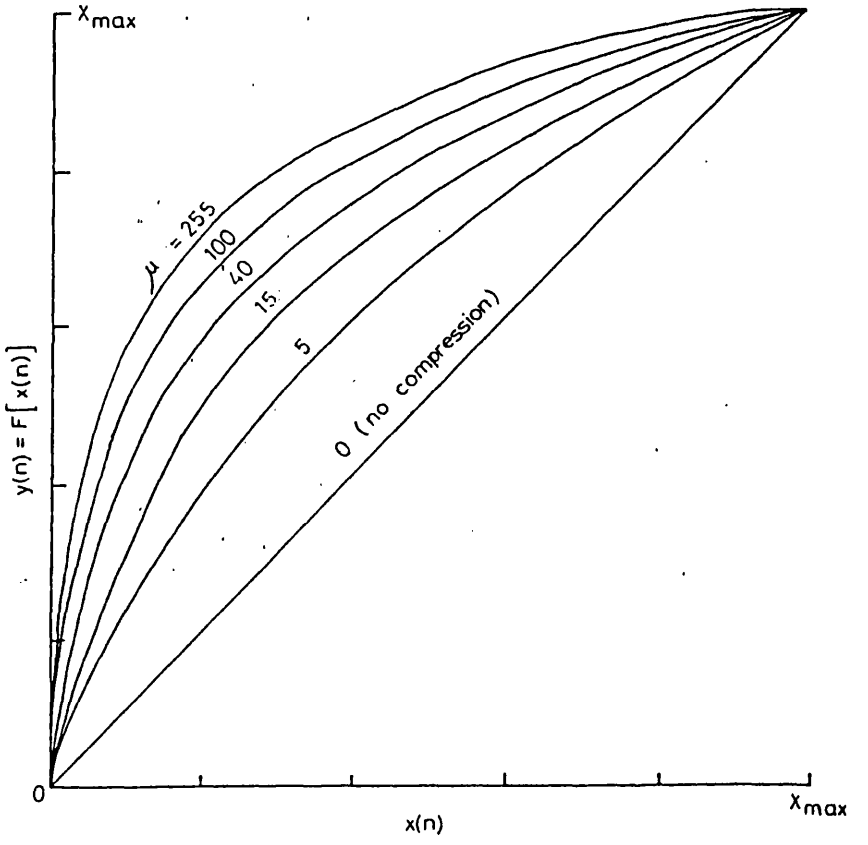


Figure 1.7 : Input - Output relationships for a μ - law characteristic. (After Smith [16]).

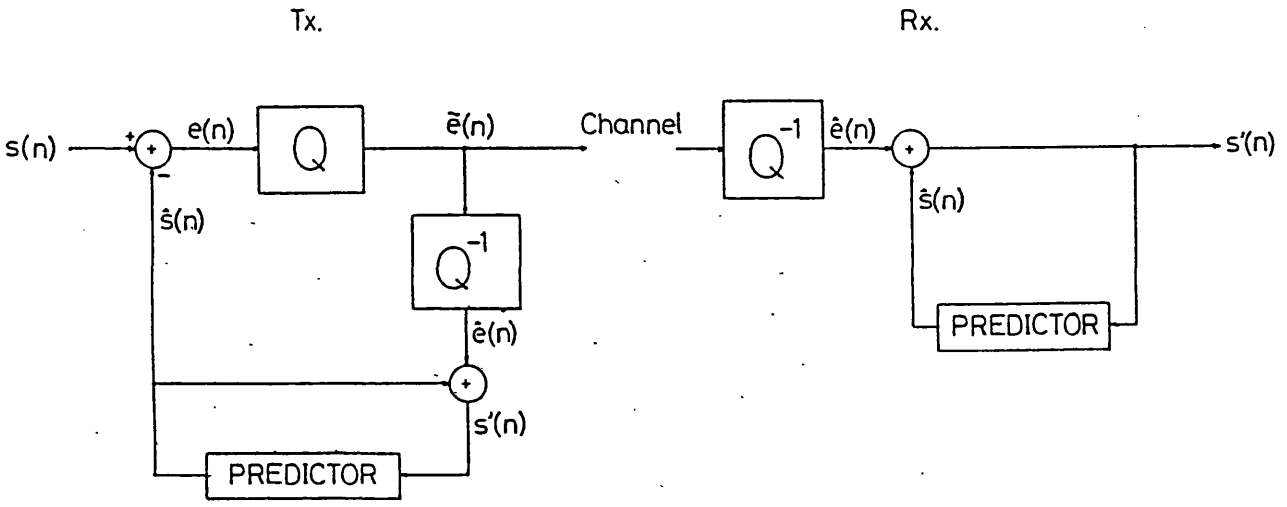


Figure 1.8 ; Block Diagram of a Differential Pulse Code Modulation(DPCM) system.

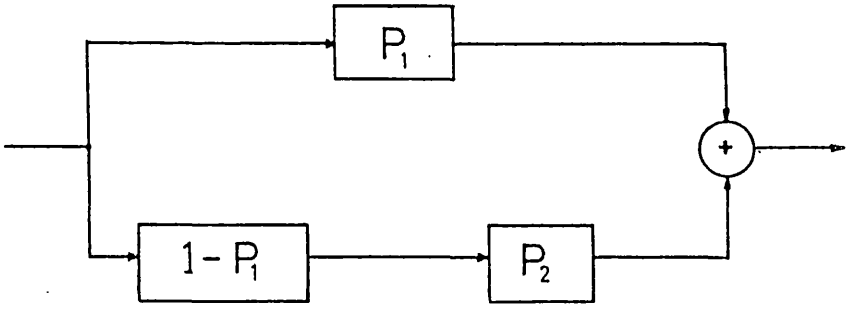


Figure 1.9 : Predictor required for Adaptive Predictive Coding(APC).

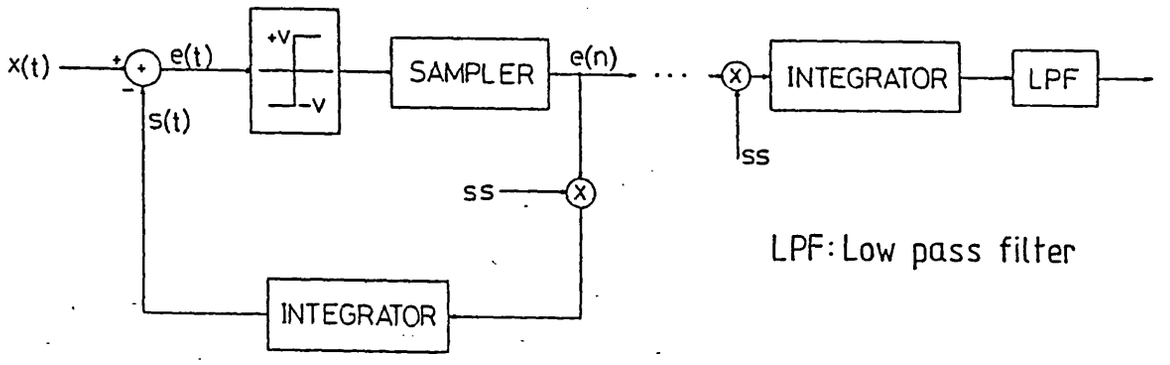


Figure 1.10 : Block Diagram of a Delta Modulation(DM) system.

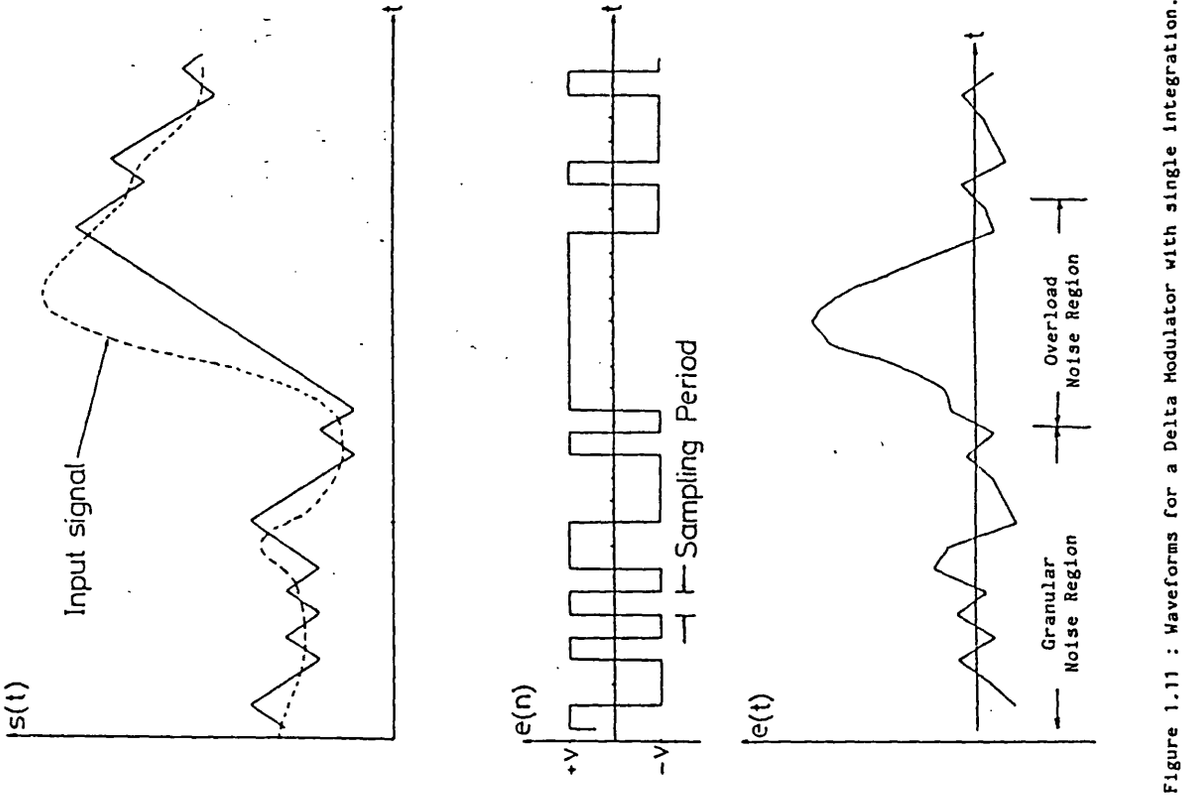


Figure 1.11 : Waveforms for a Delta Modulator with single integration.

Chapter 2

TES: Simulation and Real-Time

2.1 Introduction

The work of King and Gosling on Time Encoded Speech (TES) (section 1.4) stimulated research by Al-Doubooni [39] into speech encoding applied to low data rate transmission. The system developed by Al-Doubooni differed from that of conventional TES in the representation of the shape desciptors for successive waveform segments. Mapping of the quantised time and shape descriptors onto a reduced alphabet was excluded and amplitude information was signalled on an epoch to epoch basis rather than after every eight epochs.

Prior to sampling (f_s= 20kHz) the input speech was bandlimited to the standard telephone bandwidth (300 - 3400 Hz). The encoder then sampled and analysed the input speech to detect:

- waveform zero-crossings.
- maximum sample magnitude between zero-crossings.
- number of extrema between zero-crossings.
- number of samples between zero-crossings.

To reduce the effect of background noise and the data required for encoding periods of silence, symmetrical thresholding of the speech waveform was applied. The threshold level was set at -36dB below the peak magnitude of a complete utterance. Epochs with a peak magnitude less than the threshold level were assigned a zero valued peak magnitude, figure 2.1.1. A succession of such epochs were combined to form a single epoch of zero magnitude and duration equal to the sum of the durations of the combined epochs. To preserve signal polarity, the number of combined epochs was restricted to odd

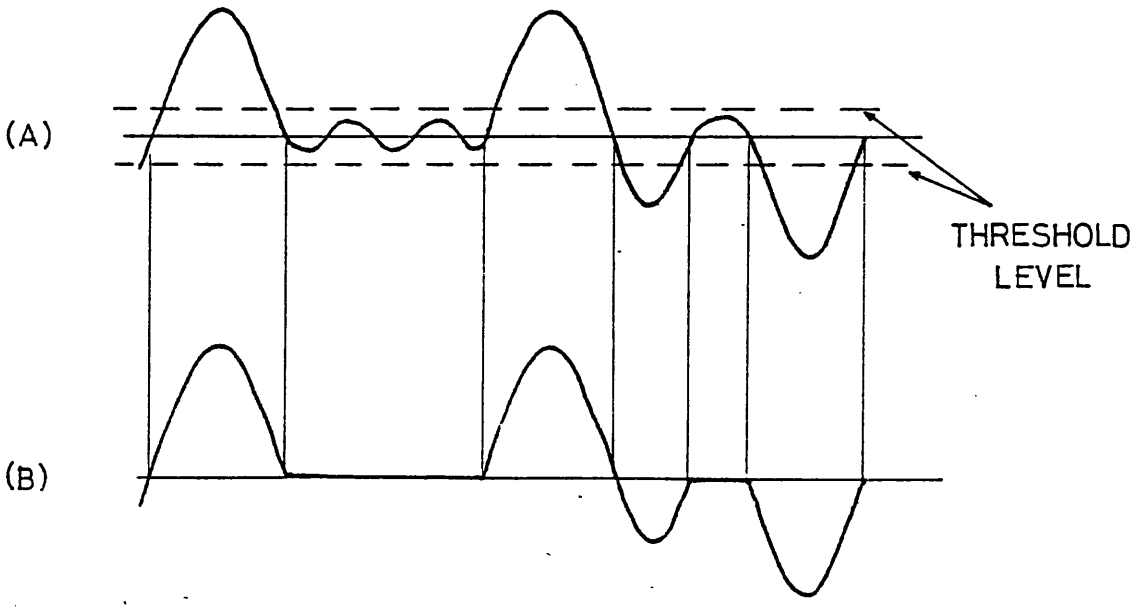


Figure 2.1.1 : Symmetrical thresholding of a speech waveform.
(A) Waveform before thresholding.
(B) Waveform after thresholding.

values. Computational limitations restricted the maximum duration of an epoch to 2047 samples (approximately 0.1 seconds). The epoch parameters 'transmitted' were:

(a) Epoch Duration
(b) Number of Extrema
(c) Epoch Peak Magnitude

Due to the technique implemented to encode periods of silence and low level noise the encoding resulted in two seperate distributions for the epoch durations. One distribution was associated with the speech and ranged from one time quantum (a time quantum being 50 μs) to 40 time quanta. The second distribution was associated with the silence intervals and ranged from one time quantum to

2047 time quanta. Since each epoch parameter was independently coded and thus had a seperate codeword alphabet the speech epoch duration codeword alphabet could be a subset of the silence epoch duration codeword alphabet. However, the total codeword alphabet required for the representation of all possible epoch parameters was vast with 2047 codewords being required to represent the epoch durations alone.

The synthesis of the speech waveform employing these epoch parameters was straight forward. Utilising a look-up table of stored epoch shapes, the epoch duration and number of extrema were employed to reference the required epoch shape. The individual samples of the referenced shape were scaled relative to the peak magnitude and then output. The synthesis algorithm assumed alternating polarity of the epoch peak magnitude. Epochs of three or more extrema were synthesised as a symmetric three extreme segment with a minima at A/2, A being the peak magnitude.

To investigate possible data reduction techniques for the transmission of TES codewords it was decided to utilise the algorithms developed by Al-Doubooni for the initial simulation exercises before progressing to implementing any techniques of significance in realtime. However, due to the vast codeword alphabets described above some adjustments to the parameters output from the encoder were required before these algorithms could be utilised. In order to be able to implement algorithms in real-time, a flexible simplex digital voice channel was developed. In section 2.2 the adjustments to the coded speech files produced by Al-Doubooni's algorithms are outlined and section 2.3 describes the simplex digital voice channel.

2.2 Simulation Investigations

Al-Doubooni's algorithms were not developed for the simulation of a feasible speech coder but as a vehicle for studying factors governing data rates and speech quality. The algorithms were implemented in non real-time on a Digital Equipment PDP 8/e minicomputer. The input speech and coded speech together with the synthesised speech were all stored on the PDP 8/e minicomputer disk storage medium.

The necessary buffering between the variable-rate source and the constant-rate channel was not simulated and therefore, distortions due to buffer underflow and overflow do not occur within the synthesised speech. The effectiveness of, and distortions introduced by, a data reduction technique could therefore be studied without having to distinguish between the distortions due to buffering and those due to the algorithm.

In a real-time system, where the input speech is bandlimited to the standard telephone bandwidth, epoch durations greater than 2ms (40 samples) are, in general, small in magnitude and infrequent. Spurious epochs of durations greater than 3.2ms (64 samples) are truncated to 3.2ms. Periods of silence and low level noise are represented by the transmission of a special codeword which represents a fixed epoch duration of 2ms and zero magnitude. The number of special codewords transmitted depends upon the duration of the silence interval or segment of low level noise. The maximum epoch duration encoded in a real-time system in therefore 3.2ms.

However, within Al-Doubooni's TES encoding algorithm the thresholding technique implemented for the coding of silence and low level noise produced epoch durations in excess of 0.1 seconds. Therefore, before the coded speech file produced by Al-Doubooni's TES encoding algorithm could be utilised its format had to be altered to resemble that produced by a 'physical' system. This was achieved by omitting all epochs whose duration exceeded 2ms and whose peak magnitude was zero. It was hypothesised that the silence codeword would not play a role in any data compression technique because it invokes a separate synthesis routine and any corruption of this codeword by a data compression algorithm could result in significant deterioration in the quality of the synthesised speech. Thus to remove such codewords from the coded speech file would not affect the results of simulation exercises.

Once data compression and expansion transformations had operated on a coded speech file, which had had the "silence" intervals removed, the synthesised speech produced using Al-Doubooni's algorithm would have been devoid of the majority of the normal conversational inter-word pauses. Therefore the capability to replace the "silence" intervals previously omitted was deemed necessary and was implemented.

Two Fortran programmes were developed. One to remove the silence intervals, SILRM.FT, and another to re-insert such intervals, SILRP.FT. SILRM.FT required the coded speech file, CSF1.DT, as its input and created a coded speech file, OUT1.DT. OUT1.DT had all those epochs with durations greater than 2ms and of zero peak magni-

tude omitted.

Once OUT1.DT had been transformed, and inverse transformed, via a data compression technique, provided the transformations had not eliminated or created extra epochs, the silence intervals were replaced by SILRP.FT. This program required two input files: the original coded speech file, CSF1.DT, and OUT1.DT or its equivalent, and created an output file, OUT2.DT.

The epoch parameters of CSF1.DT were sequentially inspected. If an epoch did not have a duration greater than 2ms and zero peak magnitude, a set of epoch parameters were transferred from OUT1.DT to OUT2.DT, otherwise the epoch parameters were transferred from CSF1.DT to OUT2.DT. Thus OUT2.DT is OUT1.DT with the silence intervals of CSF1.DT. Figure 2.2.1 depicts this entire operation.

A further program PAREXT.FT was developed for the sequential extraction, from the coded speech file, of one of the three epoch parameters. This aided the user in the production of sequential plots of the epoch parameters which were extensively used during the simulation investigations.

2.3 Real-Time Digital Voice Channel

The system described here is not a rigid solution to the problem of serial transmission of Time Encoded Speech. The overriding design criterion was flexibility and as a working basis the following properties of a comprehensive system were included:

(a) Serial transmission of binary data.

(b) Arbitrary channel bit rate.

(c) The only external synchronisation is the bit timing information.

(d) Non-propagation of channel errors.

(e) "Filler-words" as an essential element of the system.

(f) Optional inclusion of "silence-words".

(g) Optional inclusion of "repeat-words".

(h) Amplitude information included on a fixed pattern.

In line with terminology used in the majority of the literature on coding theory, "symbols" has been reserved to denote the individual members or "letters" of the underlying set used in transmission. For example, in a binary system, the symbols are 0 and 1. Therefore, the expression "tes-word" will appear in preference to "tes-symbol" and similarly for "amplitude-word","repeat-word" etc.

A simplex system was developed based on two Plessey MIPROC 16F processors. One functioned as the transmitter and the other as the receiver. A uni-directional serial interface, with an arbitrary preset constant transmission rate and the option of inserting inversion errors into the bit stream, was introduced between processors.

Section 2.3.1 describes a MIPROC processor and the software development tools used. Section 2.3.2 discusses the choice of code structure and dictionaries. Section 2.3.3 presents the read-only data structure employed whilst section 2.3.4 discusses the external

hardware and executive software. Details of the analogue interface between the input speech and transmitter, and that of the receiver and the output speech are described in section 2.3.5. Section 2.3.6 describes the digital interfaces between the two processors and section 2.3.7 describes the modular implementation of the analysis and synthesis associated with TES. The support software generated in conjuction with the system is described in section 2.3.8. Finally, the extrema and zero-crossing detection routines utilised in the example algorithm listed in appendix 1, are presented in section 2.3.9.

2.3.1 Miproc System

An overview of the complete system is presented in figure 2.2.2. The Plessey Miproc 16F [40] is a single board 16 bit processor fabricated using low power schottky TTL gates. The main features of this processor are:

- two general purpose registers.
- an index register.
- a memory address register.
- 170 possible instructions.
- execution time of 250ns for single cycle instruction.
- 64k memory addressing capability.

An extra board is required to utilise the indexed addressing mode of some of the instructions and the interrupt facility. The

Miproc architecture has completely separate data memory and program memory addressing spaces.

Two identical Miproc systems were used in the configuration of figure 2.2.2. Each system was equipped with:

- 2k of program memory.
- 6k of data memory.
- a 12 bit analogue-to-digital converter.
- a 12 bit digital-to-analogue converter.
- a dual 16 bit parallel interface.
- a 16 bit by 16 bit hardware multiplier.
- a resident monitor.

Although the hardware multiplier forms a 32 bit product in 250 ns, loading of the multiplier and multiplicand from arbitrary memory locations and the transfer of the product to arbitrary memory locations required a total of eight instructions (2 µs). Only the most significant bit of one channel of the parallel interface was employed to form the serial data link.

The Miproc resident monitor provided examine and deposit, break point, de-assemble, single step, load, dump and verify facilities. A Digital Equipment PDP 11/34 was employed to emulate the monitor console under MLINK providing the facilities for loading from and dumping to disk files created under RT-11. A cross-assembler for the convertion of programs developed in mnemonics to machine code was also available on the PDP 11/34 thus completing a powerful software development capability.

2.3.2 Code Structures and Dictionaries

(i) Code Structures

Of particular concern in a real-time implementation of TES is the irregular rate at which data is extracted from the waveform. This necessitates that some form of buffering be interposed between the irregular rate source and the constant rate channel. Since the buffer must be of finite capacity in a real-time system the possibilities of buffer underflow and buffer overflow must be considered and contingencies for such eventualities organised. If the channel rate is faster than the symbol generation rate, the buffer in the transmitter will empty. At such a time the channel must be kept occupied by the transmission of a "filler-word". The "filler-word" is formed from a special pattern of digits which are unambiguously recognised by the receiver.

To enable the inclusion of specialised codewords it was assumed that the serial data stream would comprise of a succession of elements. Each element being either a filler-word or data frame. Each data frame may be either a specialised codeword, such as the silence- or repeat-word, or an amplitude-word followed by a group of N tes-words. Figure 2.2.3 illustrates possible data frames.

To further retain flexibility in the system, separate dictionaries were implemented so that the amplitude-dictionary was the totality of amplitude-words and the tes-dictionary the totality of tes-words. Certain special cases may arise when conditions are placed on these dictionaries eg. When the amplitude-dictionary is

empty and the system is "amplitudeless" as in the same sense as that of infinite clipping. Alternatively, it may be that the amplitude--dictionary is a proper subset of the tes-dictionary.

Since no conditions were placed on the construction of the two dictionaries, they were considered disjointed and not necessarily having the same structure eg. constant length codewords in the amplitude-dictionary and variable length codewords in the tes-dictionary. Such a structure had provisions for combining the dictionaries or omitting the amplitude-dictionary in special applications.

It was decided that the silence-word would be decoded as if it were a single descriptor for a group of N epochs of fixed duration and zero amplitude. It therefore replaced the standard data frame of "amplitude-word followed by a group of N tes-words". If a member of the amplitude-dictionary was reserved as the silence-word, the receiver recognised its presence and inserted a group of zero amplitude epochs. Similar remarks apply to the inclusion of a repeat-word.

Since the filler-word was extracted from the bit stream on every occurrence and was always preceded by either another filler-word or a data frame, it also provided the minimum synchronisation requirement of the channel. The filler-word was constrained to occur only at the beginning (end) of a data frame and was not a member of either the amplitude- or tes-dictionary. Therefore, the construction of the two dictionaries and filler-word were not independent. For efficient synchronisation in the presence of channel errors it was necessary to augment the naturally occurring filler-

words (due to transmitter buffer underflow) with deliberately inserted filler-words at a set rate. Figure 2.3(a) depicts the codeword classifications, while figure 2.3(b) provides an example of possible dictionaries where the amplitude-dictionary is based upon a 5 bit constant length code and the tes-dictionary is based upon a 4 bit constant length code.

(ii) Code Dictionaries

The code structure selected was, in general, one in which the same codeword could have two different meanings, depending on its position within the data frame. Therefore, correctly identifying position within the data frame was essential. Indeed it was felt that the importance of maintaining the correct frame boundaries outweighed that of conventional error protection for guaranteeing the data within the frame. Because of this, the vast bulk of linear algebraic coding techniques [41-43] were not considered.

Constant length codes had the attractive property that once synchronisation was achieved, simply counting the number of bits provided for continuing synchronisation. Methods of synchronising constant length codes have been suggested by many authors. Usually these methods require special coding of the message, as in comma-free codes [44-47] or the periodic insertion of a comma between blocks of messages [48,49]. For application to small dictionaries, comma-free coding has an inherently large redundancy. The word "comma" is interpreted as a sequence of symbols which cannot occur in any valid sequence of data-words. The latter approach was anticipated in section

2.3.2(i) when it was stated that the filler-word could provide the minimum synchronisation. The overall redundancy in this approach (including the bits needed for synchronisation) increases with the frequency of transmission of the comma and is therefore, to a certain extent, under the control of the transmitter.

The self-synchronising properties of some variable length codes were not universally applicable in the system under development due to the structure imposed upon the sequence of codewords. For such codes, decoding from an arbitrary starting point in a given sequence eventually results in the correct word endings being identified [50]. However, without further additions or modifications, such a code cannot identify the frame boundaries and hence a possible change of dictionary. Furthermore, a transmission error may result not only in the decoding of the wrong codewords but even the wrong number of codewords, which would also affect frame boundary identification.

Given a finite dictionary with a maximum length for its codes, there exists a finite number of variable length codes. A procedure for constructing dictionaries which minimise the average number of code symbols required to encode a "message" was proposed by Huffman [51]. Similarly, codes can be constructed for minimising the probability of buffer overflow where the said buffer is necessarily interposed between the codeword source and the channel. Constant length codes cannot be applied to achieve such optimisations. Minimisation of the expected delay for matching the constant rate channel to the variable rate source not only requires a knowledge of the isolated word probabilities but also of their sequential statistics. There-

fore, the optimisation procedures which generalise Huffman's algorithm cannot be applied here because they only utilise the long-term codeword probabilities. However, this does not affect the ability of variable length codes to reduce, if not minimise, such functions.

The implementation of the digital interfaces developed (section 2.3.6) were to be equally suited (ie. employed the same assembler routine) to constant and variable length codes given some minor constaints on allowable dictionaries.

2.3.3 Read-Only Data Structures

The processing time required for a real-time TES implementation was obviously at a premium. It was therefore the function of the read-only data structures, commonly termed 'look-up tables', to reduce to a minimum any repetitive calculations and data manipulations. By performing all the necessary calculations beforehand and storing the results, the only processing overheads were those of addressing the relevant portion of the look-up table. These overheads were related to the amount of structure built into the look-up tables. The final composition of real-time calculation and look-up operations was a compromise between two options:

(1) the extravagant use of processing time for repeated calculations.

or (2) the complexity of addressing diverse look-up functions.

It was clear that at some stage the limitations of data memory

size would play a role. However, initially the view that data memory was cheap and readily available was adopted.

Two principal gains were made by identifying those parts of the processing which would benefit from the application of look-up tables and then defining a precise format for such tables. Firstly, the preparation of the tables was made semi-automatic and secondly, the low-level coding was simplified and standardised. Since the preparation of the tables was without reference to any actual memory location, they were relocatable and therefore regarded as physical Read-Only Memory (ROM) to be loaded anywhere within the data memory address space.

An example to illustrate the above points is given in figure 2.4. The purpose of this table is to convert amplitude information as established within the transmitter into a corresponding transmitter buffer entry. The look-up table contains entries, correctly ordered, for each possible amplitude value and the first entry is used to verify that the amplitude legal value is legal.

The format of the decoding ROM for amplitude codewords is shown in figure 2.5. The two arrays contained constants required in the decoding algorithm (SUM1(k) = $\sum_1(k)$, SUM2(k) = $\sum_2(k)$ of appendix 2). The application of the variable length decoding algorithm with the ROM of figure 2.5 was considered superior to the simple "code book" approach as the latter would occupy an enormous amount of memory for the codewords envisaged (up to 12 bits).

Figure 2.6 gives the format of the ROM for the encoding of tes-

words. The ROM is essentially a two dimensional matrix stored by columns. The first two locations of the table were used to verify the legality of the row and column numbers. The column offsets were pre-calculated and related to the start of the ROM.

The format of the ROM employed to decode the tes-words and to store the look-up shape for waveform synthesis is given in figure 2.7. To distinguish one look-up shape from another and minimise computation, it was necessary to record the length, in samples, of each shape.

2.3.4 External Hardware and Executive Software

Each program had one analogue and one digital interface separated from the main analysis and synthesis sections by first-in first-out (FIFO) buffers. Associated with each interface was an interrupt service routine (ISR). The analogue interface was assigned higher priority than that of the digital interface. In order to minimise sampling "jitter" the analogue interrupt service routine was permitted, as far as was possible, to interrupt the digital interrupt service routine. Both sampling and transmission were controlled by external clocks.

With the boards available, the most sensible means of transferring data between the Miproc processors was via one channel of the parallel interface card (the serial interface cards were restricted in their formats and dedicated to servicing the monitor terminals). By driving the interface cards interrupt generation circuitry

at a known rate and transferring, under software control, one bit on a single line of the external bus it was possible to continuously change the bit rate of the serial link. If timing overheads associated with the interrupt service routine proved too great, it was then possible for up to 16 of the external bus lines to be utilised and, provided the bit-rate calculations were suitably ammended and the order and singularity of the data bits respected, it would remain valid to talk of a serial transmission channel.

To avoid programming complexities it transpired that it was sensible to devote the most significant line(s) of the external bus for data transfers. Once the appropriate word had been output to the control register of the transmitter's interface card, an interrupt was generated in Miproc 1 when "DATA STROBE IN" went low. The software responded, if the interrupt was unmasked, by executing the transmitter interrupt service routine. Similar remarks apply to Miproc 2. Thus by connecting the "DATA STROBE OUT" port of the transmitter (which goes low whenever data is put on the external bus) to the "DATA STROBE IN" port of the receiver, the required synchronisation of bit timings between processors was achieved. Thus the transmission rate clock was input to the "DATA STROBE IN" of channel A of the transmitter parallel interface and channel B of the parallel interfaces acted as the serial link between transmitter and receiver. An outline of the external hardware necessary for the interface is given in figure 2.8.

The executive software performed the following functions:

- set up hardware interfaces.

- initialisation of program variables and buffers.
- control of analysis and synthesis routines.
- processed fatal error conditions.

The transmitter's executive software ensured that the analysis continued as long as the input buffer (which contained signal samples) contained data. Conversely, the receiver's executive software ensured that the synthesis continued as long as the output buffer was not full. In the receiver, during system initialisation, dummy codes were inserted into the receiver buffer. These codes represented a tone which was synthesised and repeated until synchronisation was achieved, at which point they were overwritten by received data.

Any error conditions occurring during the execution of either program was considered fatal causing a diagnostic message to be output to the monitor terminal followed by a halt in program execution. Figure 2.9 gives the flowchart of the executive software for both transmitter (Tx.) and receiver (Rx.).

2.3.5 Analogue Interfaces

The analogue interfaces (figure 2.10) were straightforward. Prior to sampling, the input speech was bandlimited to the standard telephone bandwidth (300 to 3400 Hz) and the synthesised output speech samples were low-pass filtered with a cut-off frquency at 3400 Hz.

In the transmitter, analogue-to-digital conversions (ADC) of the input speech were triggered by the falling edge of an external

clock. The converter's "end of convert" signal caused an interrupt which was serviced by the analogue interface software. The ISR allowed a constant value to be subtracted from the converted signal immediately prior to storage. This ensured that the analysed signal had zero mean. After writing to the sample buffer, a test for imminent buffer overflow was conducted which indicated whether the processor was capable of processing the samples fast enough. If the buffer were to overflow then a fatal error condition was generated.

In the receiver, the analogue ISR performed a test to ensure that the FIFO containing the synthesised samples was not empty. An empty buffer would have occurred if the output samples could not be computed fast enough and resulted in a fatal error condition. However, if a sample was available in the buffer, it was read and output to the digital-to-analogue converter (DAC) and then the interrupt latch was cleared.

Both routines were clocked at the same frequency, usually 10, 15 or 20kHz depending on the complexity of the main program.

2.3.6 Digital Interfaces

The digital interface software was more involved than that of the analogue interfaces. This was a consequence of the fact that the filler-word may well be up to three Miproc 16 bit words long (depending upon the data frame employed).

In order to accommodate variable length, as well as constant length codes, the buffer structure characterised in figure 2.11(a)

was employed. The 16 bit words stored in the buffer were split into two fields. The most significant 4 bits (count field) indicated the number of valid bits in the least significant 12 bits (data field). There was therefore a limit of 12 bits to the size of the data word and a conventional restriction that the most significant bit of the data occupied bit 11 of the buffer. By utilising such a buffer between the TES analysis and the serial channel, all possible data types were treated alike by the bit orientated transmission routine.

A flowchart of the transmitter digital interface is given in figure 2.12. Transmission of a single bit occurred in response to an externally generated clock signal (section 2.3.4). The ISR associated with the transmission buffer retained a count of how many bits of the current word were left to transmit. If there were no words in the buffer, a filler-word was inserted into the bit stream. The test for an empty buffer involved comparing the read and write pointer values. It was conceivable that the ISR might interrupt that portion of the transmitter program which updated the write pointer. To prevent this causing any problems, the pointer update was performed after a complete data frame had been transcribed to the buffer. The filler-words introduced for the purpose of additional synchronisation passed straight through the transmitter buffer.

Figure 2.11(b) shows the actual transfer of the data stored in the buffer of figure 2.11(a) for the situations where one, two and four external bus lines are used to form the "serial" link (see section 2.3.4). The short dash (-) indicates data-word boundaries.

The receiver ISR had to do more than merely take bits off the

channel and buffer them individually, it was also required to achieve de-framing of the bit stream into its elements. It therefore had to acknowledge the presence of filler-words and be able to distinguish the various species of data-word.

Since filler-words were removed before they entered the receiver buffer the ISR introduced a delay equal, in bits, to the length of the filler-word. For ease of programming this length was constrained to a multiple of 16. The bit stream was continuously scanned for the filler-word (synchronisation-word) and re-synchronised on its every occurrence. This prevented erroneous synchronisation from persisting indefinitely.

At any instant the receiver might be out of synchronisation either because it had been turned on at an arbitrary time or because an error had caused loss of synchronisation. One bit of the system status flag was reserved to indicate the presence or absence of synchronisation. Once synchronised, the receiver must identify from which dictionary decoding is to take place. This was accomplished by utilising another bit of the status flag. If this bit was set the next word was drawn from the amplitude-dictionary and if clear from the tes-dictionary.

The receiver digital interface flowcharts are given in figures 2.13(a),(b) and (c). The receiver's ISR was equally well suited to decoding constant length and variable length codes. A bit count was continually compared with the value of the maximum number of bits per word (a value stored within the decoding ROMs). A fatal error occurred if the bit count exceeded this limit, otherwise the decoding

continued. The 'ceiling' values were pre-calculated (see appendix 2) and stored within Decoding Array 2 of the decoding ROM (figures 2.5 and 2.7).

To retain flexibility of data frame structure, counters were incorporated to assess the number of amplitude and tes-words decoded. This made it possible to transmit more than one amplitude-word, a feature which might be desirable in a special case.

2.3.7 Analysis and Synthesis Modules

The processing involved in analysis and synthesis was sub-divided into three 'subroutines':

(1) that associated with individual samples

(2) that associated with epochs

(3) that associated with complete data frames (ie. N epochs)

Within the transmitter (analyser) the sample routine had to read a sample from the ADC buffer, update the descriptive parameters of the current epoch and check whether an "end of epoch" had been reached. This was indicated by a change in sign of the input signal. The epoch routine had to validate, encode and reset the descriptive parameters of the previous epoch, update envelope information and check whether the data frame was complete. The frame routine validated, encoded and reset the envelope parameter(s) and transcribed the complete data frame to the transmission buffer. In periods of high data generation rate, the transmission buffer may not have room for the complete data frame. It was therefore the responsibility of

the frame routine to test for imminent buffer overflow and act accordingly. The overflow prevention stratergy had to respect the frame structure of the data within the transmission buffer. To this end, it proved useful to maintain a record of the transmission buffer write pointer which indicated the beginnings of the last M frames. These values were stored in a 'discard' buffer, a location of which was overwritten every time a new frame was processed. The length of this FIFO buffer, M, determined the amount of data, in frames, to be discarded. In the event of imminent transmission buffer overflow, the write pointer of the transmission buffer was overwritten with the current output from the 'discard' buffer. This effectively back spaced the write pointer and discarded M complete data frames. The current data frame was then written to the transmission buffer. In addition to, the basic frame processing, the transmitter retained a count of the number of frames transmitted and at regular intervals, in terms of complete data frames, inserted a filler-word into the data stream. This provided a synchronisation facility to augment the built in dependance on 'buffer underflow' filler-words.

Similar remarks to the above can be made for the receiver software. The sample routine computed the next sample by multiplying the correct look-up value by the envelope factor and stored the result in a FIFO buffer ready for output by the analogue ISR. The sample routine accessed the epoch processing routine if the end of the synthesised epoch was reached. The epoch routine decoded the variables which ensured that the correct look-up shape was addressed, reversed the polarity of the scaling factor and at, the end of the data frame, accessed the frame processing routine. The frame pro-

cessing routine handled imminent receiver buffer underflow and decoded the amplitude-word(s). Once again, it proved useful to maintain a record of the receiver buffer write pointer which indicated the beginnings of the last M frames. These were stored in a 'repeat' buffer, a location of which was overwritten every time a new frame was received. The length of this buffer determined the amount of data, in frames, that was to be repeated. In the event of imminent receiver buffer underflow, the receiver buffer read pointer was overwritten by the current output from the repeat buffer. This effectively backspaced the read pointer causing the previous M data frames to be repeated.

To reduce the perceptually disturbing effects which result when a series of epochs are repeatedly synthesised, each time the read pointer was backspaced, the amplitude-word(s) at that location, if greater than zero, were decremented. Therefore, following a series of repeats, the amplitude-word(s) may have been reduced to zero resulting in 'silence' being synthesised.

Figures 2.14(a),(b) and (c) give the transmitter and receiver sample, epoch and frame processing routines.

2.3.8 Software Support

Once the read-only data structures had been defined (section 2.3.3) a suite of Fortran programmes were developed by P. S. Cooper [52] to run on the PDP 11/34. These generated ROM files according to parameters set interactively.

The amplitude encoding and decoding ROM's were produced using a program which required inputs to specify the number of possible amplitude levels, the degree of compression (in the case of logarithmic characteristics or zero for a linear characteristic), the number of codewords and their lengths (n_ks of appendix 2). The program constructed the actual codewords and computed the relevant buffer entry (count field and data field) for each possible input level. Figure 2.15 demonstrates program/user interaction required to create a 7 bit logarithmic ($\mu = 255$) quantiser. The user responses are underlined.

The corresponding program for generating the TES encoding and decoding ROM's required data to specify the minimum and maximum epoch lengths (dimension 1) in samples, the minimum and maximum number of extrema (dimension 2), the number of codewords and their lengths (n_ks of appendix 2). For each codeword the user specified minimum and maximum values of dimensions 1 and 2 and the synthesis shape. Figure 2.16 demonstrates the inputs required to create a six bit (63 codeword) linear epoch encoder with:

(a) minimum value of dimension 1 : 1

(b) maximum value of dimension 1 : 64

(c) minimum value of dimension 2 : 1

(d) maximum value of dimension 2 : 6

The synthesis shapes were stored individually in disk files and were created and edited by further Fortran programmes. Figure 2.17 gives an example of one possible TES-coding matrix for an alphabet of 39 codewords which could be implemented using the program

described above.

2.3.9 Extrema and Zero-Crossing Detection

The listing of appendix 1 gives the code developed to transmit, in real-time, a TES data frame which comprises of one n-bit amplitude-word (maximum of $2^n - 1$ codes) per N m-bit tes-words (maximum of $2^m - 1$ codes), where $n + N.m$ is less than 49. Silence- and repeat-words were not incorporated into this particular coding but the provision for extra synchronisation- (filler-) words was included. The object of this section is to describe the techniques employed in this implementation for the detection of extrema and zero-crossings.

(i) Extrema Detection

An extreme was defined as occurring when the gradient between successive samples changed sign. Mathews [53] demonstrated that extrema which were small, in comparison with the local topography, may be omitted without causing a perceptible difference to the synthesised speech.

The technique employed for extreme detection and the zero-crossing detection were similar. The previous input sample, S1, was subtracted from the current input sample, S0, to yield the gradient between samples, DS0. If the absolute value of DS0 was greater than or equal to the extreme sensitivity measure, DSMIN, the previous gradient, DS1 (whose absolute value was greater than or equal to DSMIN) and DS0 were compared for a sign change. If DS0 was less

than DSMIN no comparison was performed. Once a sign change was detected the extrema count was incremented. Figure 2.18 portrays the extreme detection and small extrema elimination routine.

(ii) Zero-Crossing Detection

A "zero-crossing" was initially defined as two successive samples having different polarity. For the purpose of this definition zero was conventionally regarded as having a positive sign, being consistent with the 2's complement binary representation of the system ADC's and DAC's.

However, in noisy environments, uncertainty of zero-crossing positions resulted in the generation of epochs which were short in duration and low levelled in amplitude compared with the local topography. In turn, this caused an increase in the occupancy of the transmission buffer and hence increased the possibility of transmission buffer overflow. At the receiver, short epochs manifested as a "hissing" in the background of the synthesised speech.

To reduce this effect, a sample sensitivity measure, SMIN, was introduced. If the absolute value of the current sample, S0, was greater than or equal to SMIN, then the sign of SS1, ie. the previous sample whose absolute value was greater than or equal to SMIN, and the current sample were compared to determine if a sign change occurred. However, if the current sample value was less than SMIN no comparison was made. If a sign change was detected the 'end of epoch' routine was entered. Figure 2.19 depicts the operation of

the zero-crossing detector and demonstrates how it reduces the number of short epochs due to noisy environments.

2.4 Summary

This chapter reviewed the simulation algorithm developed by Al-Doubooni for studying Time Encoded Speech (TES). A description of alterations to the coded speech files output by this TES coder were indicated. These alterations were necessary before the studies presented in chapters 4, 5 and 6 could be conducted.

Also presented was a description of a versatile real-time digital voice channel developed for the implementation of TES coders. The choice of code structures and code dictionaries, analogue and digital interfaces, external hardware, executive software and the analysis and synthesis modules were detailed. The final section presented one solution for the detection of the extrema and zero-crossings of the speech signal. The application of this simplex digital voice link are presented in chapter 7.

Coded Speech File 1.
(CSF1)

2047	0	0	1
2047	0	0	1
1986	0	0	1
7	0	9	-1
13	0	0	1
6	0	9	1
24	0	0	1
9	0	13	-1
7	0	14	1
3	0	0	1
6	0	12	1
7	0	15	-1
54	0	0	1
7	0	12	1
8	0	15	-1
49	0	0	1
11	0	9	-1
7	0	9	1
24	0	19	-1

Coded Speech File 2.
(CSF2)

7	0	9	-1
13	0	0	1
6	0	9	1
24	0	0	1
9	0	13	-1
7	0	14	1
3	0	0	1
6	0	12	1
7	0	15	-1
7	0	12	1
8	0	15	-1
11	0	9	-1
7	0	9	1
24	0	19	-1

CSF2 CSF3 CSF3A

CSF1 CSF3 CSF3A

2047	0	0	1	7	0	9	-1	2047	0	0	1
2047	0	0	1	7	0	0	1	2047	0	0	1
1986	0	0	1	13	0	9	1	1986	0	0	1
7	0	9	-1	9	0	0	1	7	0	9	-1
13	0	0	1	9	0	13	-1	7	0	0	1
6	0	9	1	7	0	14	1	13	0	9	1
24	0	0	1	6	0	0	1	9	0	0	1
9	0	13	-1	6	0	12	1	9	0	13	-1
7	0	14	1	7	0	15	-1	7	0	14	1
3	0	0	1	7	0	12	1	6	0	0	1
6	0	12	1	8	0	15	-1	6	0	12	1
7	0	15	-1	7	0	9	-1	7	0	15	-1
54	0	0	1	7	0	9	1	54	0	0	1
7	0	12	1	24	0	19	-1	7	0	12	1
8	0	15	-1					8	0	15	-1
49	0	0	1					49	0	0	1
11	0	9	-1					7	0	9	-1
7	0	9	1					7	0	9	1
24	0	19	-1					24	0	19	-1

Figure 2.2.1 : Shows the operation of silence removal (SILRM) from the coded speech files and the re-insertion of the silence (SILRP) into the coded speech files

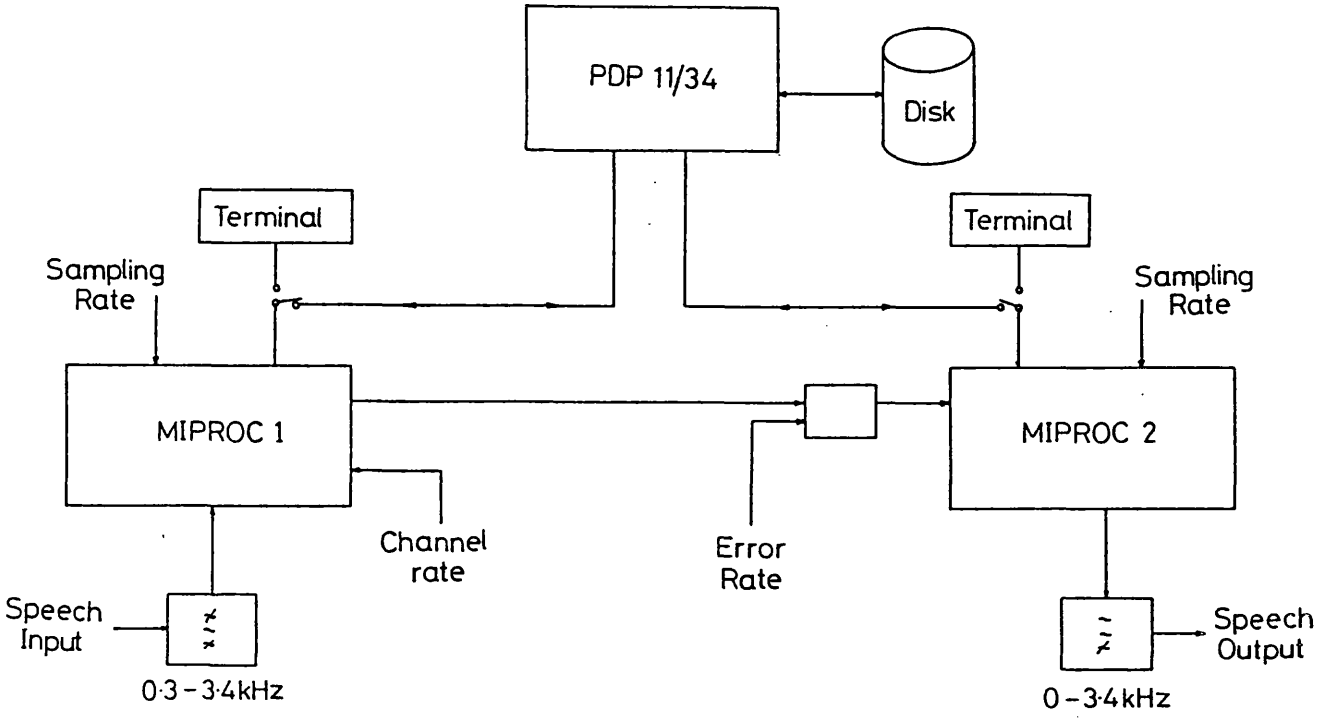


Figure 2.2.2 : System overview.

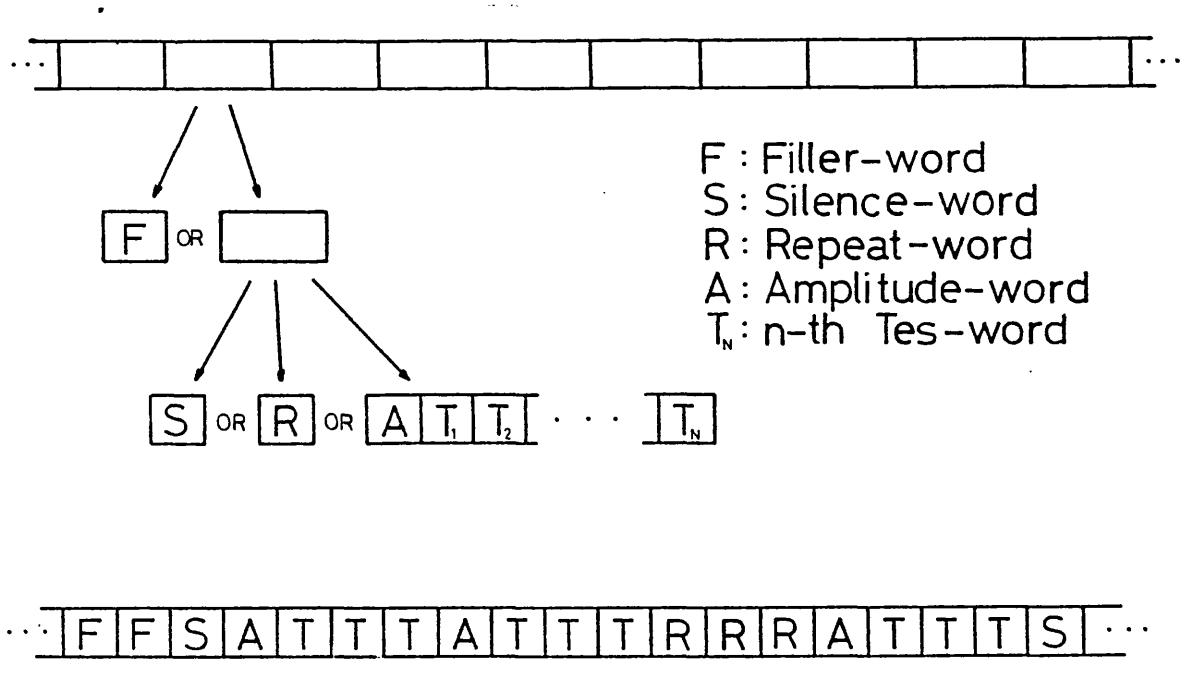


Figure 2.2.3 : Illustration of the possible data frames.

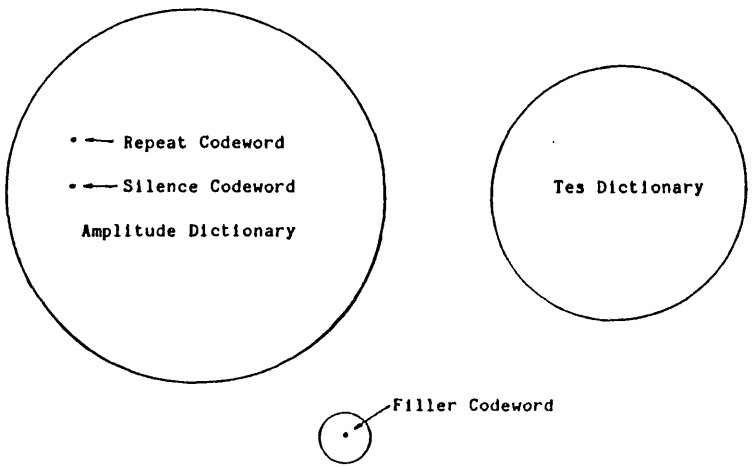


Figure 2.3(a) : Codeword Classification

Amplitude Dictionary

0 0 0 0 0	Silence Codeword
0 0 0 0 1	Repeat Codeword
0 0 0 1 0	
.	
.	
.	
1 1 1 0 1	
1 1 1 1 0	

Tes Dictionary

0 0 0 0
0 0 0 1
.
.
.
1 1 0 1
1 1 1 0

Filler Codeword

1 1 1 1 1 0

Figure 2.3(b) : Example of 5 bit constant length Amplitude Dictionary and 4 bit constant length Tes Dictionary.

LOOK-UP SIZE	Number of Rows : NR
	Number of Columns : NC
OFFSETS	Column(1) = 2 + NC
	. . .
	Column(i) = 2 + NC + (i-1).NR
	. . .
	Column(NC) = 2 + NC + (NC-1).NR
Column 1	Tes-Word(1,1)
	Tes-Word(2,1)
	. . .
	Tes-Word(NR,1)
Column 2	Tes-Word(1,2)
	Tes-Word(2,2)
	. . .
	Tes-Word(NR,2)
. . .	
Column NC	Tes-Word(1,NC)
	Tes-Word(2,NC)
	. . .
	Tes-Word(NR,NC)

Figure 2.6 : Tes Encoding ROM

Look-Up Size	Number of Input levels, N
Codewords	Amplitude Codeword (1)
	Amplitude Codeword (2)
	Amplitude Codeword (3)
	. . .
	Amplitude Codeword (n)

Figure 2.4 : Amplitude Encoding ROM

Dictionary Size	Number of Codewords, N
Word size	Maximum Number bits/word, K
Decoding Array 1	Sum1(1)
	Sum1(2)
	. . .
	Sum1(K)
Decoding Array 2	Sum2(1)
	Sum2(2)
	. . .
	Sum2(K)
Output Levels	Amplitude(1)
	Amplitude(2)
	. . .
	Amplitude(N)

Figure 2.5 : Amplitude Decoding ROM

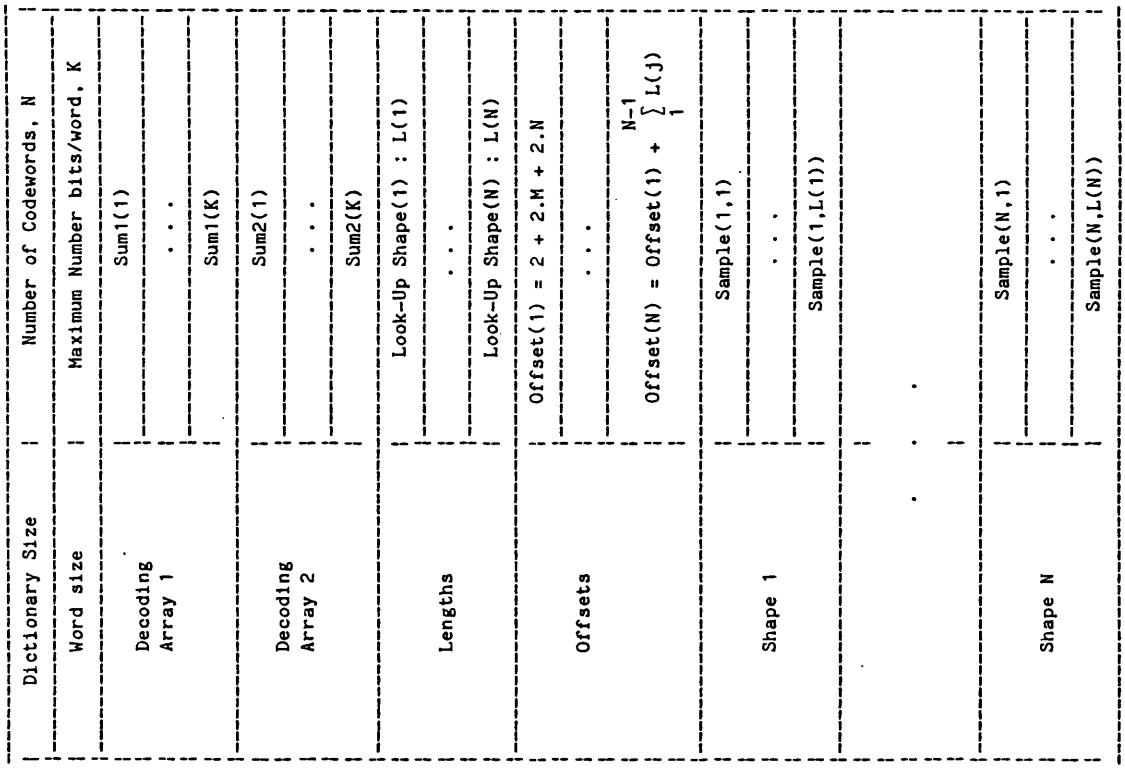


Figure 2.7 : Tes Decoding and Waveform Reconstruction ROM

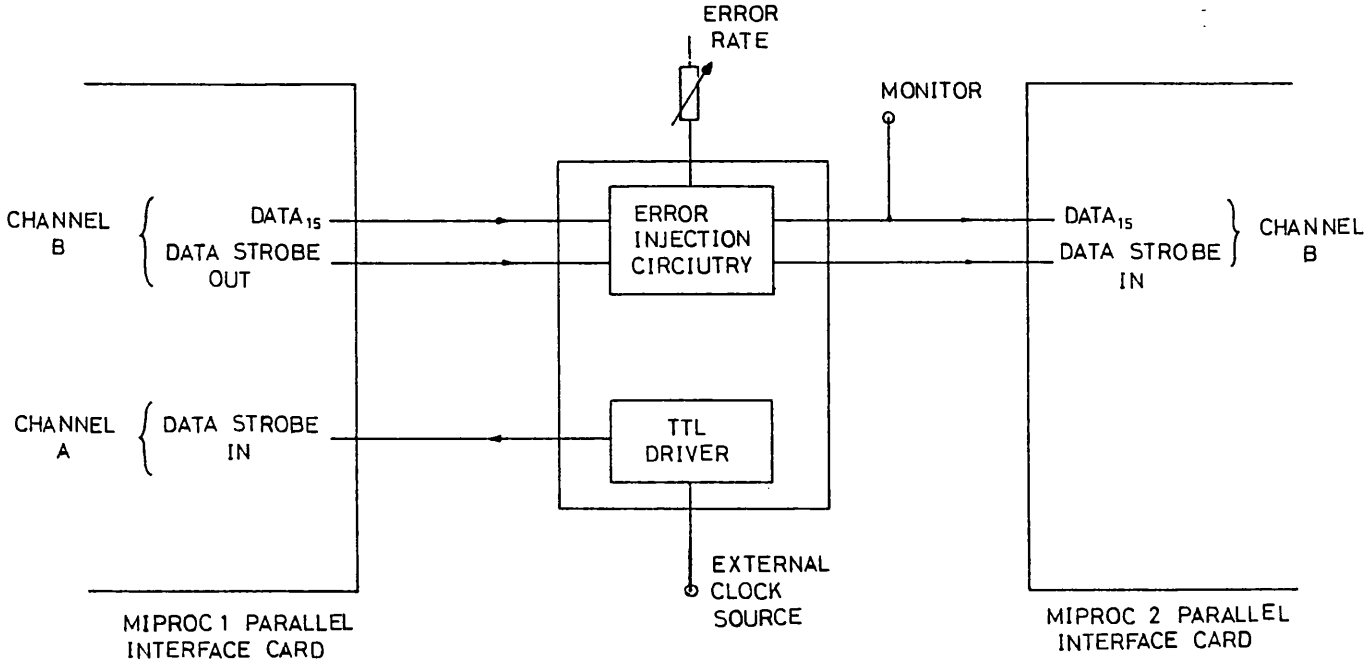


Figure 2.8 : External Hardware.

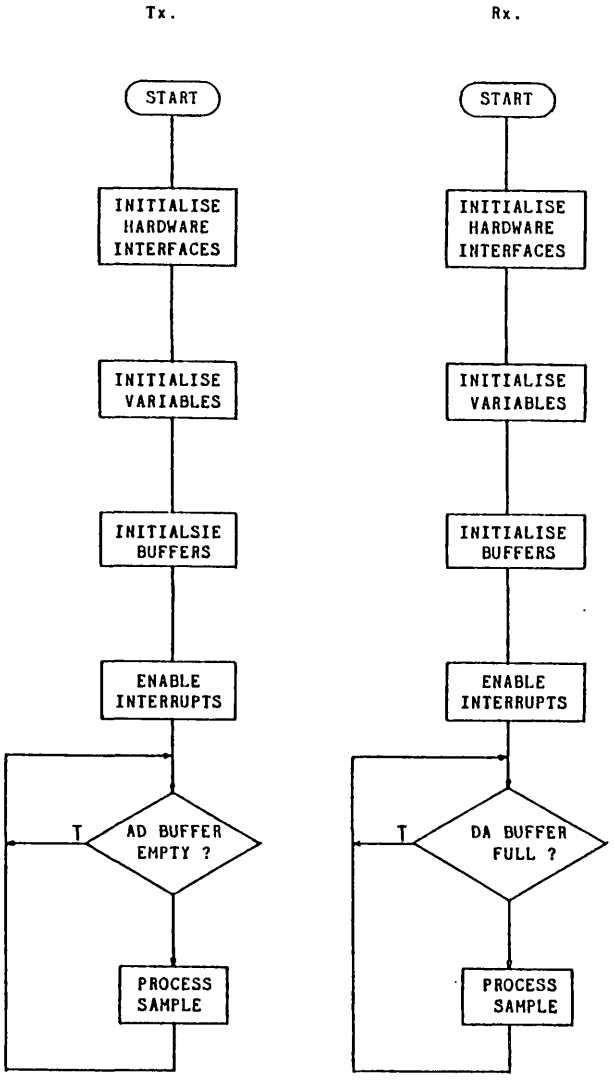


Figure 2.9 : Executive Software

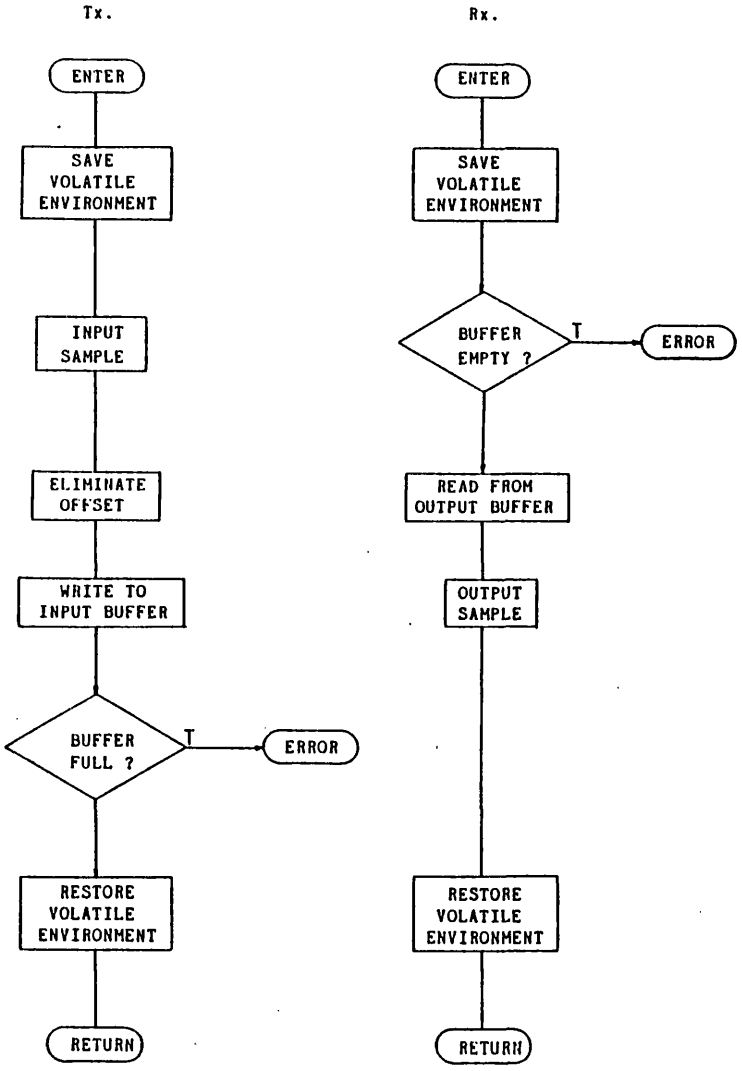


Figure 2.10 : Analogue Interface

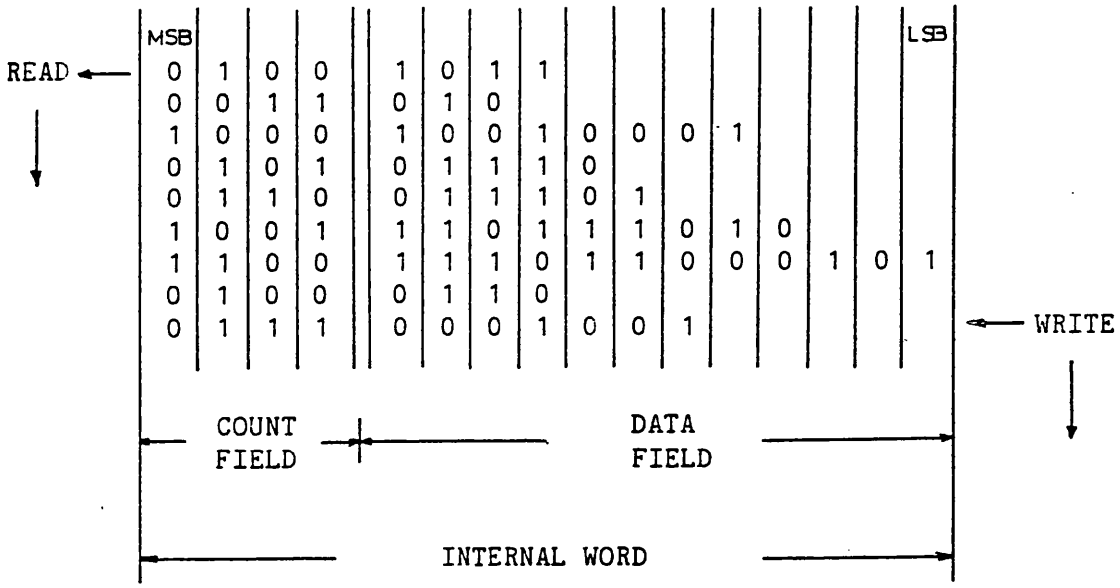


Figure 2.11(a) : Transmission Buffer Structure

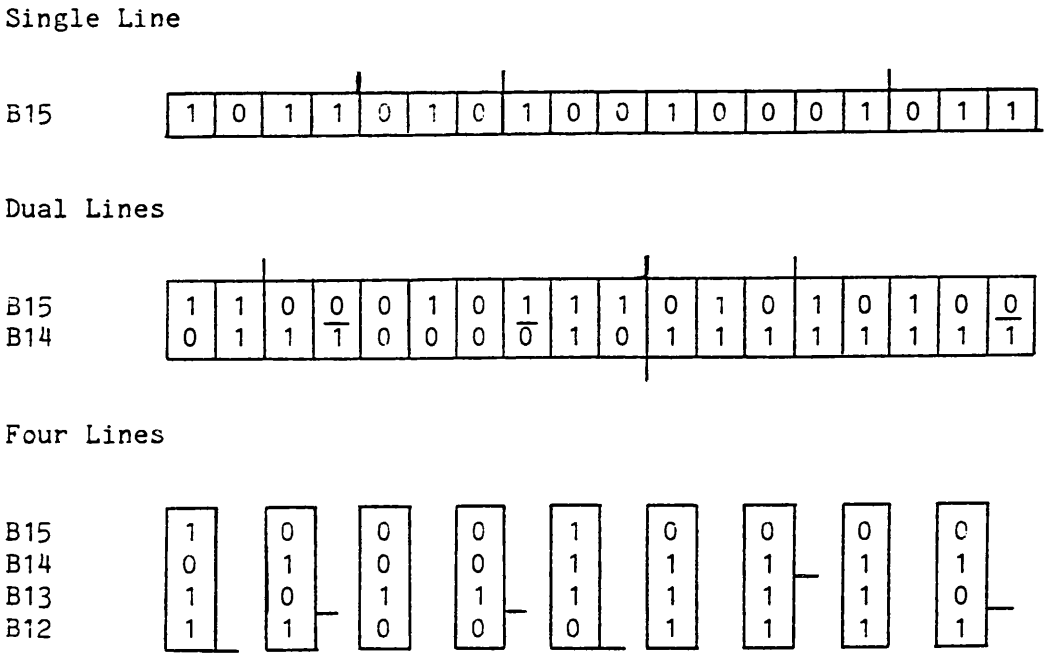


Figure 2.11(b) : Transfer of data stored in figure 2.11(a) employing 1, 2 and 4 external bus lines.

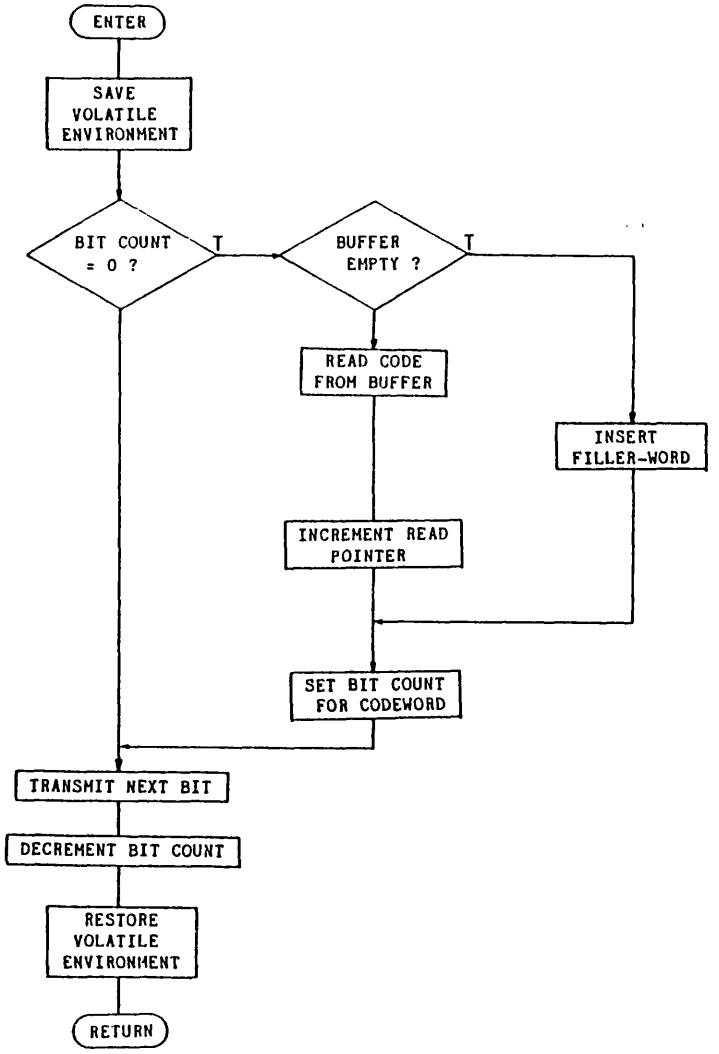


Figure 2.12 : Transmitter Digital Interface

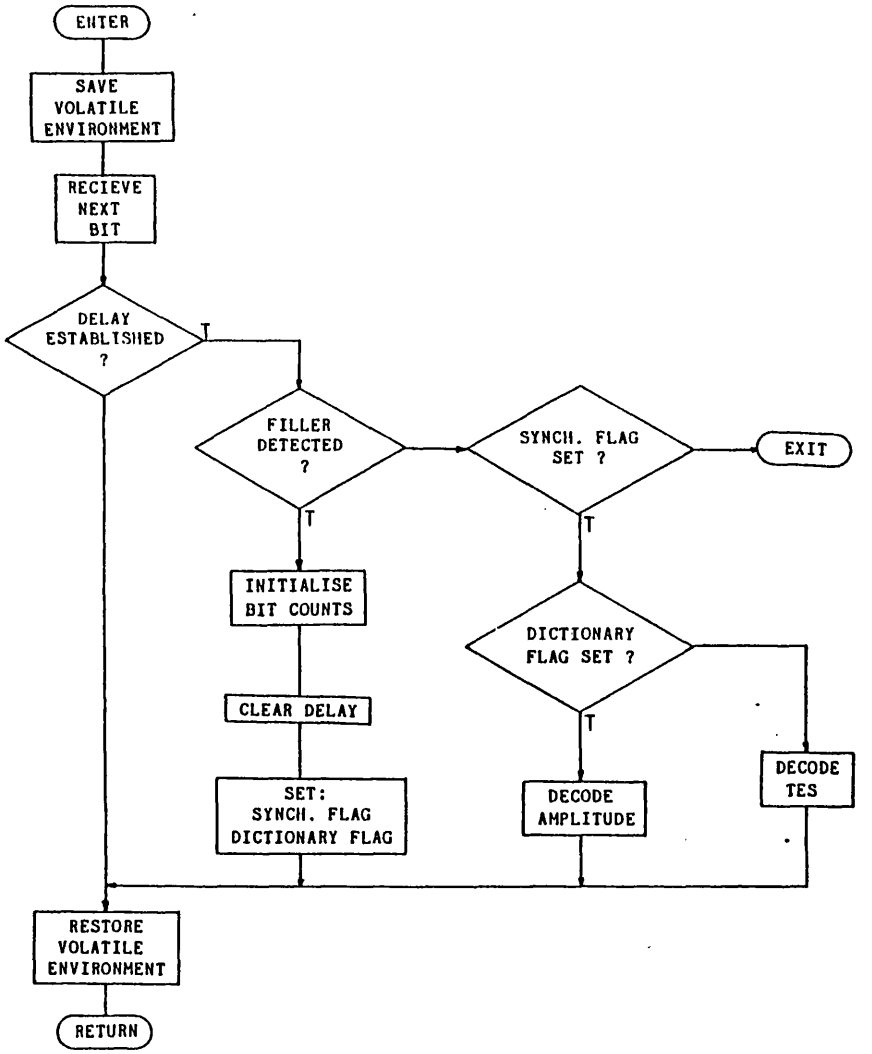


FIGURE 2.13(a) : Receiver Digital Interface - Synchronisation and Control.

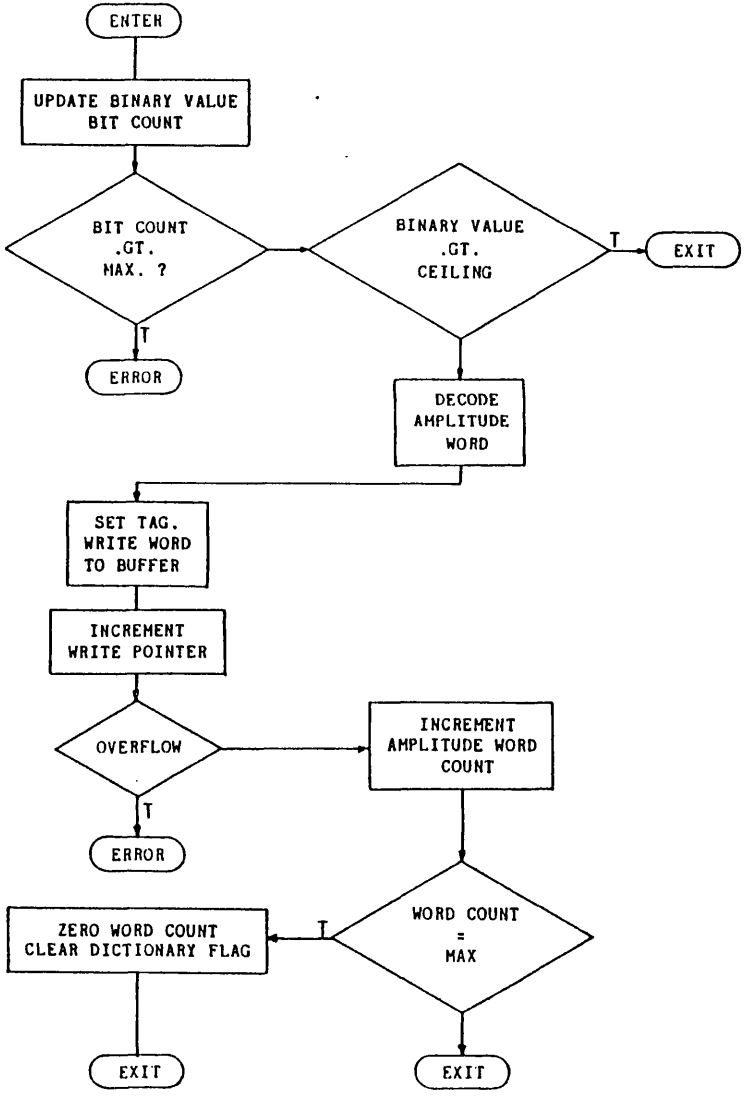


Figure 2.13(b) : Receiver Amplitude-bit Decoding

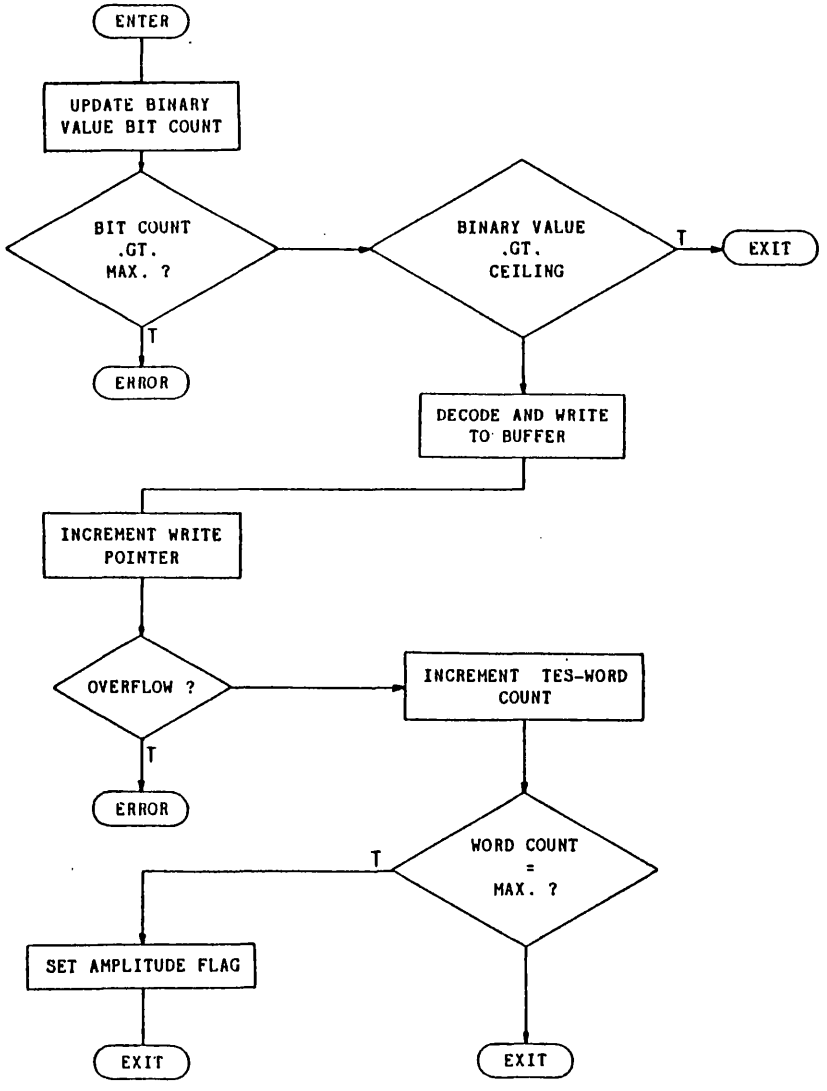
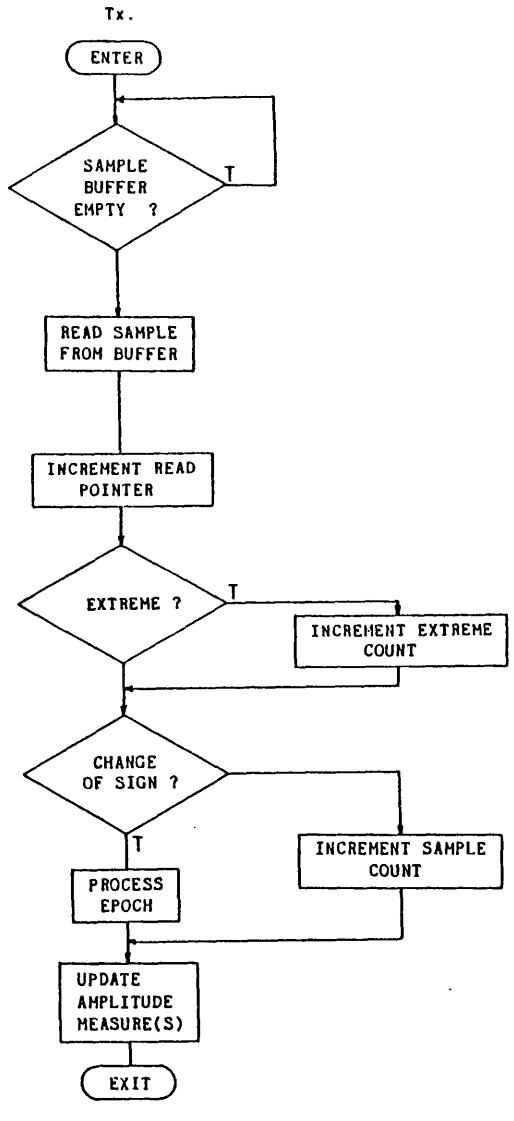


Figure 2.13(c) : Receiver Tes-bit Decoding



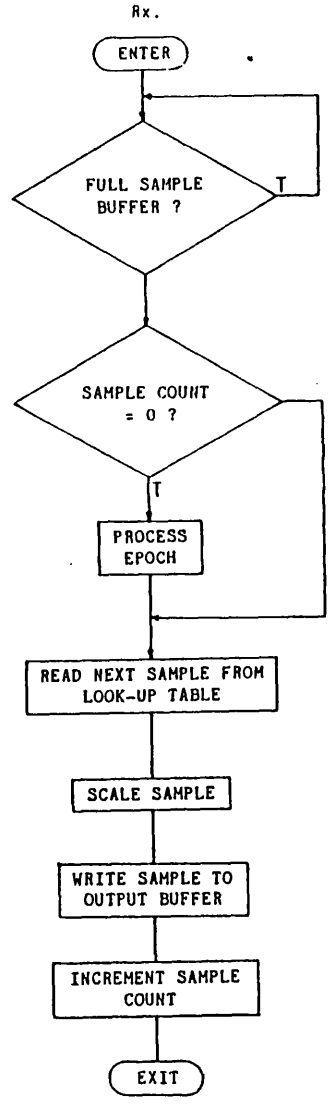
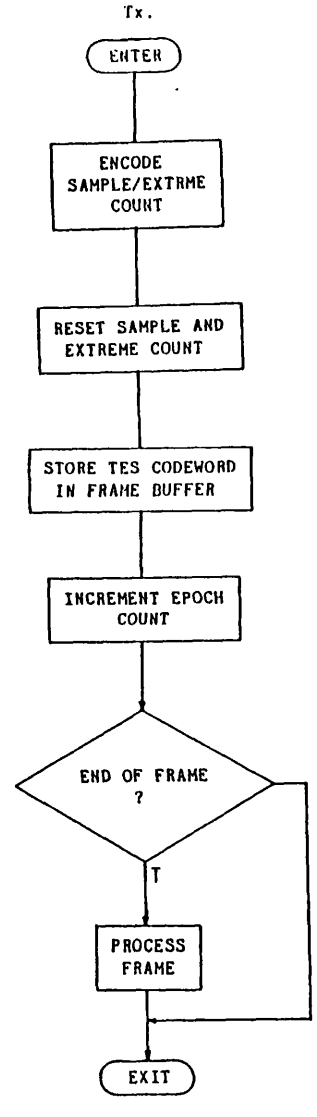


Figure 2.14(a) : Sample Processing



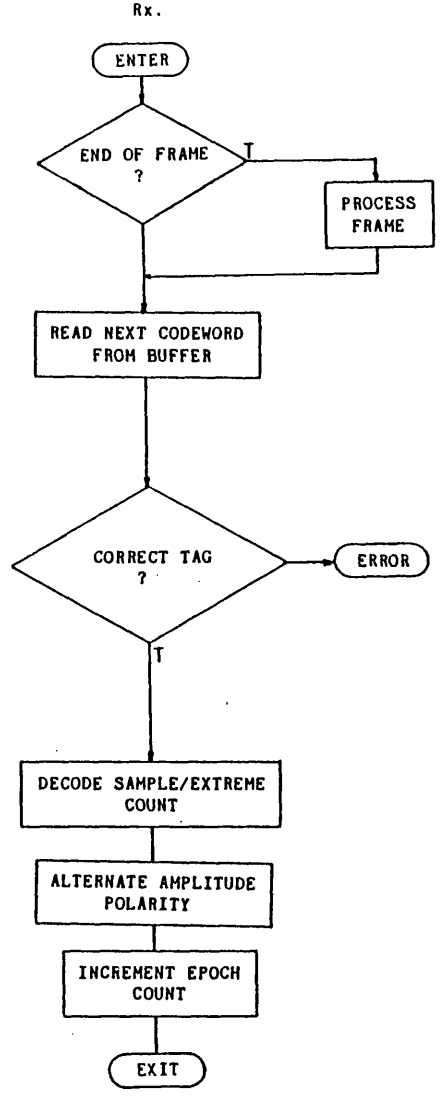


Figure 2.14(b) : Epoch Processing

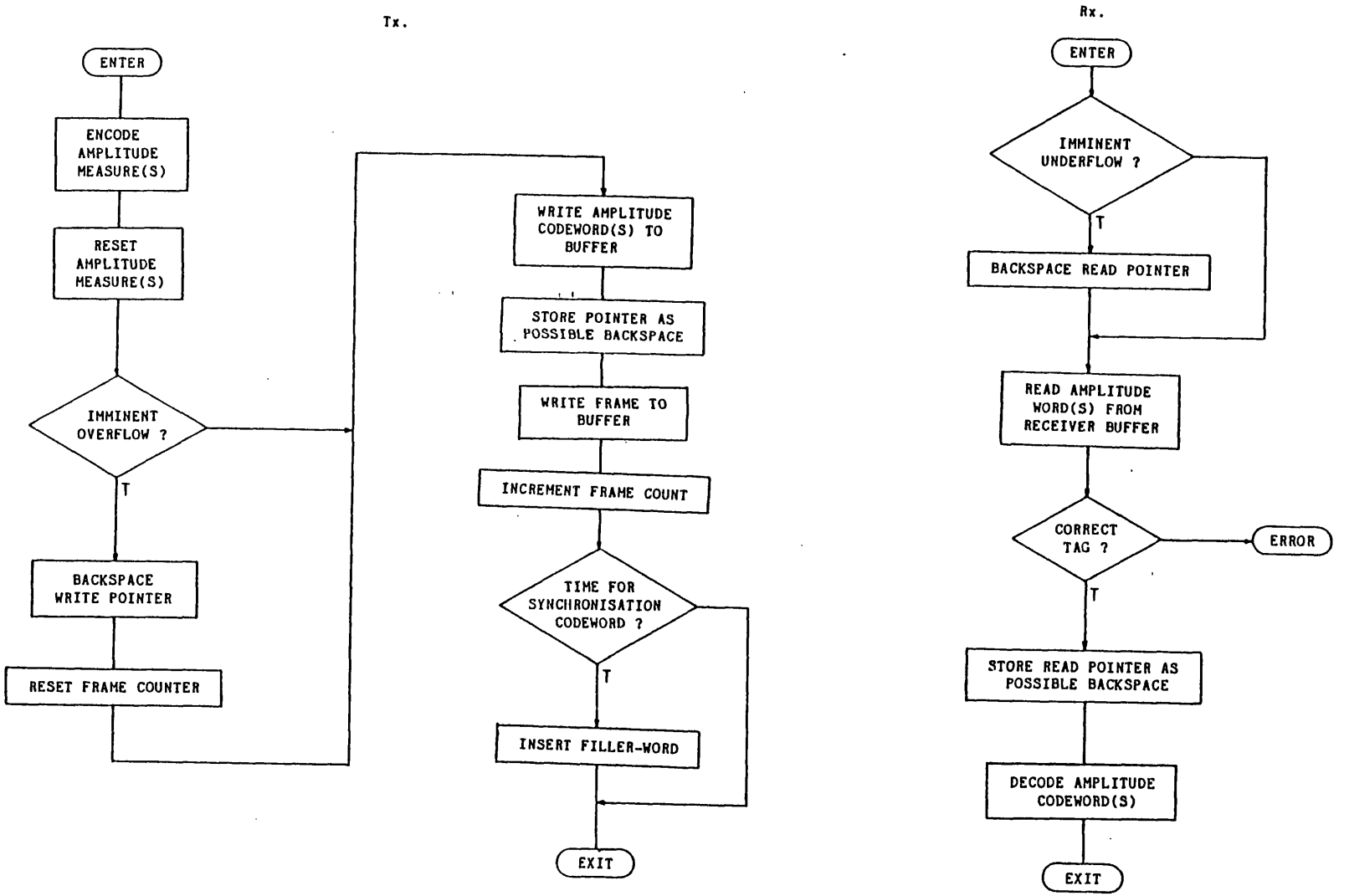


Figure 2.14(c) Frame Processing Routine

Encoding Information Filename : AENCOD

Decoding Information Filename : ADECOD

Number of input levels : 2047

Degree of Compression : 255

Number of codewords : 127

Number of codewords of length 1 : 0
Number of codewords of length 2 : 0
Number of codewords of length 3 : 0
Number of codewords of length 4 : 0
Number of codewords of length 5 : 0
Number of codewords of length 6 : 0
Number of codewords of length 7 : 127

- Stop -

Figure 2.15 : Program/User interaction required for the construction of amplitude encoding and decoding files.

```
Encoding Information Filename :  EENCOD

Decoding Information Filename :  EDECOD

          Dimension 1 minimum :  1
                      maximum :  64

          Dimension 2 minimum :  1
                      maximum :  6

          Number of codewords :  127


Number of codewords of length 1 :  0
Number of codewords of length 2 :  0
Number of codewords of length 3 :  0
Number of codewords of length 4 :  0
Number of codewords of length 5 :  0
Number of codewords of length 6 :  63

Codeword Number 1

          Dimension 1 minimum :  1
                      maximum :  3

          Dimension 2 minimum :  1
                      maximum :  6

        Epoch Definition file :  EP2

                  Full Scale :  1

                     Correct ?  Y

                     .
                     .
                     .

Codeword Number 63

          Dimension 1 minimum :  62
                      maximum :  64

          Dimension 2 minimum :  4
                      maximum :  6

        Epoch Definition file :  EP63

                  Full Scale :  1

                     Correct ?  Y


Transcribing matrix to file

Copying Look-up shape

- Stop -
```

Figure 2.16 : Program/User interaction required for the construction of Tes encoding and decoding files.

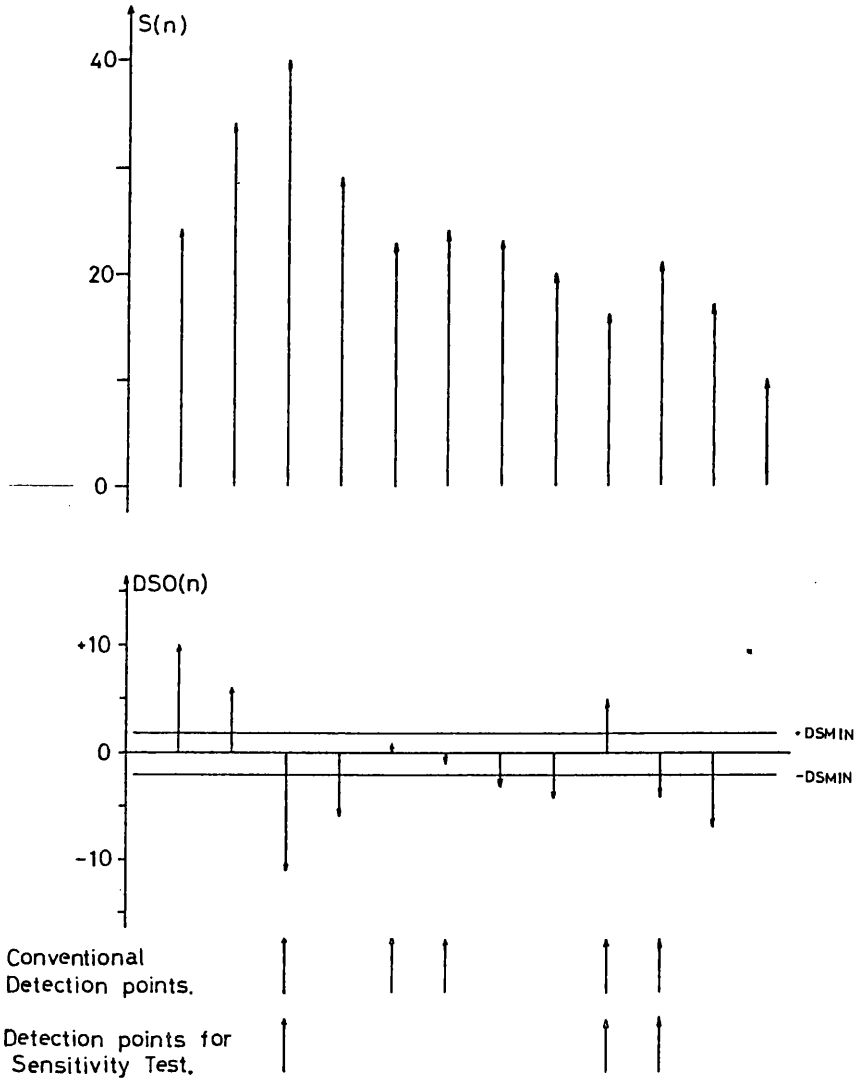


Figure 2.18 : Extreme Detection incorporating extreme sensitivity test.

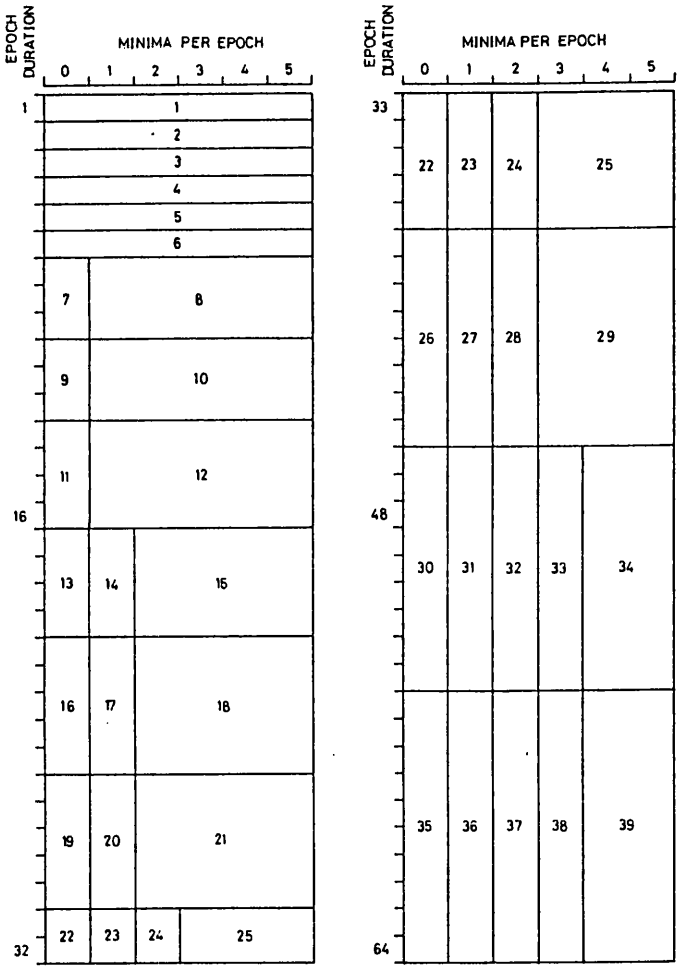


Figure 2.17 : Example of possible Tes-coding Matrix for an alphabet of 39 codewords.

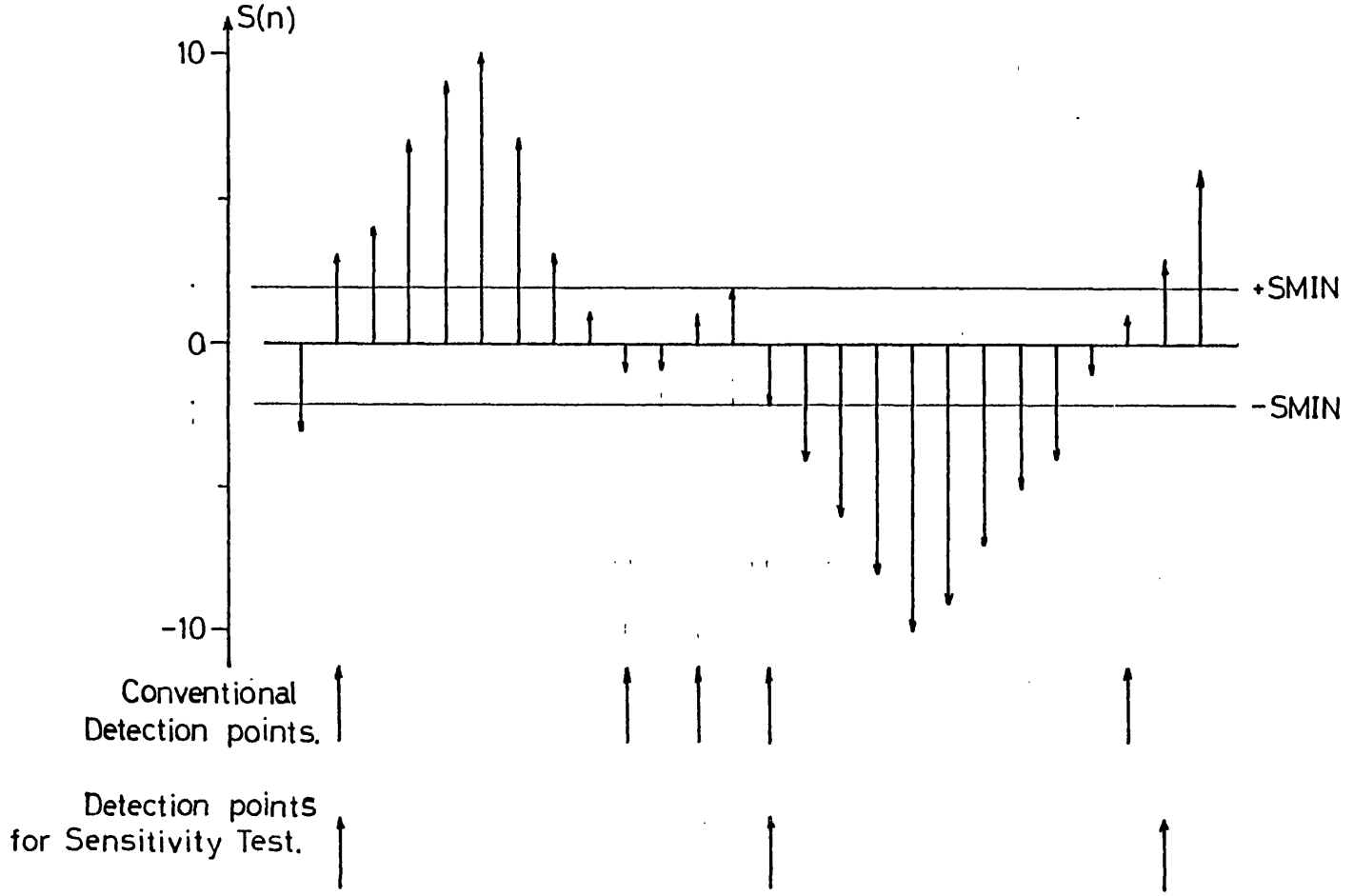


Figure 2.19 : Zero Crossing Detection incorporating sample sensitivity test.

Chapter 3

Amplitude Coding

3.1 Introduction

Licklider [54] investigated the effects of different types of amplitude distortion on the intelligibility and quality of speech. The investigations examined the techniques of peak clipping (symmetric, asymmetric and infinite) and centre clipping of the speech signal. The conclusions reached indicated that amplitude distortions degrade the "quality" of speech more severely than they do the "intelligibility". In further experiments Licklider and Pollack [32] investigated the effects of differentiation, integration and infinite peak clipping upon the intelligibility of speech. Depending on the listeners skill and familiarity with the test words, for infinite clipping of speech waveforms random word intelligibility scores of at least 70% were achieved. However, the resultant speech was of poor quality. When the infinite clipping was preceded by differentiation the intelligibility scores were as high as 90% for an "unpracticed" listener and upto 97% after the listener had become familiar with the vocabulary and with the effects of distortions.

From this and other work [33,55 - 57] it would appear that the instantaneous amplitude of speech carries little information. However, with high degrees of amplitude compression noise between words becomes overbearing and obtrusive such that the speech loses its naturalness. It is observed that the mean amplitude of speech waveforms is a relatively slowly varying characteristic. When developing TES King and Gosling (Chapter 1, section 1.4) elected to exploit this feature and transmit a mean amplitude signal at intervals. Generally, amplitude signalling produces only a very modest increase in the

data rate but in return yields a significant increase in speech naturalness. Al-Doubooni [39] investigated the problem of signalling epoch peak magnitude for TES and, on an epoch to epoch basis, concluded that "one or two bits for differential amplitude signalling was to be the most suitable for a low data rate system". Al-Doubooni's routines were not fully described: apart from stating the adaptive relationship investigated, step sizes employed for the 1 bit and 2 bit differential amplitude signalling were not specified. Furthermore, in the system studied by Al-Doubooni the peak amplitude of each epoch was transmitted utilising the full range of the quantiser (9 bits) and although high quality speech was achieved by this means in simulation, such an extravagant utilisation of data is worthy of further investigation especially as no evaluation of quality versus amplitude information per epoch was made.

Although it was believed that the amplitude signalling employed by King and Gosling was adequate, the question of coding efficiency versus quality remained unresolved. The question remaining in this area of TES implementation was: "Could the coding efficiency be increased without causing further speech quality degradation or might this possibly even improve the speech quality?"

The objectives of the investigations to be reported were:

(a) To examine possible data reduction techniques as suitable candidates for amplitude signalling within a TES system.

(b) To establish the effects of amplitude signalling in the TES domain on the quality and intelligibilty of speech.

(c) To establish a minimum amplitude information requirement for good quality intelligible speech.

An advantage of Time Encoded Speech (TES) when compared with other coders is its relatively simple implementation. The investigations into amplitude coding techniques to be reported here were conducted with the view of future implementation within a TES system hence, throughout the algorithms development, certain restrictions were imposed upon a number of features such as coder complexity and system delay. Any significant increases in these parameters would reduce the overall viability of TES.

The investigations to be described were conducted in real-time using a Plessey Miproc 16F processor (see section 2.3.1). The format of this chapter is as follows: Section 3.2 outlines the techniques investigated. Section 3.3 presents an informal subjective appraisal of the speech output and section 3.4 gives the results of the performance assessments. The conclusions of these investigations are contained in section 3.5.

3.2 Amplitude Processing

The input speech was bandlimited (0.3 to 3.4kHz) using an eighth order Barr and Stroud variable filter and Pulse Code Modulated (PCM), sampling at 20kHz. Since all processing was conducted between interrupts generated by the Analogue-to-Digital Converter (ADC), the maximum processing time per sample was 50 μs (200 single instruction cycles of the Miproc 16F processor).

During these investigations it was necessary to prevent the distortion of epoch features (other than the amplitude parameter) in order to study the effect of amplitude signalling on the quality and intelligibility of speech. To eliminate the distortions introduced by buffer overflow and reduced mapping (quantisation of epoch duration and 'shape') the individual Pulse Code Modulation (PCM) samples, x(n), were input to a First In First Out (FIFO) buffer. At the output of the FIFO the samples were processed (scaled), depending upon the particular amplitude parameter being processed, to yield a new set of samples, x'(n).

By utilising a FIFO buffer the processing retained the original relative shape of the epoch and the values of the samples relative to the peak amplitude of the epoch. This differs from conventional TES where the output speech is synthesised from stylized waveshapes. As stated briefly above, this process also avoids the quantisation errors which occur when the encoder performs reduced mapping of the epoch duration and waveshape (numbers of extrema). Since the TES codewords are not generated, there is no need for buffering which is usually associated with the constant-to-variable rate source. This means that distortions due to buffer overflow are avoided.

An important point to note here is that the peak amplitude changes were performed coherently at every zero-crossing or after a group of zero-crossings such that there were no sharp discontinuities being created in the signal.

The algorithms were implemented such that, after parameter and variable initialisation, the processor idled until an interrupt was

generated by the Analogue-to-Digital Converter (ADC). This caused the processor to execute an interrupt service routine which contained all output, input and sample analysis code.

The extent of processing required per sample varied depending on whether the amplitude parameter was updated or an 'end of epoch' condition occurred. Thus, if the result generated by the interrupt service routine (ISR) was output at the end of the current routine, the time period between samples output would have been non-uniform resulting in speech waveform distortions. To prevent such occurrences, the results were delayed until the next ISR and output at the beginning of that routine.

The value of the d.c. offset introduced by the system hardware was determined before processing commenced and the constant OFFSET was initialised to this value. On entering an ISR the current ADC output was read from which the constant OFFSET was subtracted which eliminated the systems d.c. offset. The result of this operation was stored in a register while the result of the previous ISR was output. From the time of entry into the ISR to the time of output of the previous results 1.5 μs (6 single instruction cycles) elapsed. This time period was constant for all entries into the ISR and therefore the period between samples output was a constant 50 μs. Figure 3.1 depicts how uniform sample output was achieved (DAOUT(1)) when the result was delayed and output at the beginning of the next ISR rather than that achieved when 'end of ISR' processing was employed (DAOUT(2)).

Programmes were developed for the processing of one of three

amplitude parameters :

1) Epoch peak amplitude

2) Epoch mean amplitude

3) Epoch r.m.s value

In the processes where all three measures were directly comparable no perceptible differences could be heard. This is partially in agreement with the findings of Phillips and Thomas [58] who conducted informal listening tests and concluded that adjusting the signal peak amplitude to a preset level produced the best results and the most acceptable quality. The author found this to be the situation only if the mean or r.m.s value of the epoch was amplified to the same level as that previously employed for scaling the peak amplitude. The peak amplitude is, by definition, greater than the mean value of an epoch. Therefore, if the mean value is scaled to a level previously employed for peak amplitude scaling, severe peak clipping of the mean value scaled waveform will be experienced. As demonstrated by Licklider and Pollack [32], this will produce speech which is intelligible but of poor quality. Similar remarks apply if the r.m.s value was scaled to a level previously utilised for peak amplitude scaling.

Although a fast algorithm was developed for the calculation of the epoch r.m.s value, it suffered severe timing limitations. These limitations arose from the need for 32 bit register manipulation in the summation of the squared input samples, divisions for calculating the mean of the squares and the square root subroutine.

Therefore the peak amplitude measure was preferable to the mean

and r.m.s measures from a computational stand point and, in the interest of simplicity, was employed exclusively in the subsequent investigations.

In view of the fact that a minimum bit-rate system was the overall goal of the work it was decided to commence the investigations by studying the effects of the exclusion of all amplitude information in the synthesis process. As the investigations progressed so the quantity of amplitude information utilised was increased and the resulting effects upon speech quality and intelligibility were studied.

It has been assumed that the peak amplitude of the incoming signal is represented by the maximum sample value. However, a sample may not coincide with the peak and there will be an error introduced. To reduce this error the bandlimited speech was over sampled at 20kHz which is 2.5 times greater than the required Nyquist sampling rate. Therefore the inaccuracies introduced by the assumption are significantly reduced. The original work reported by King and Gosling [30] was also conducted using a 20kHz sampling rate. Therefore the results of these investigations will be applicable to the coders developed to date.

3.2.1 Algorithm Representation

Conventional flow charting has a number of weaknesses as a working document for representing assembly language programming. To adequately present the algorithms developed during the investigations

to be reported, the author developed an alternative method of representation which has been termed a SYSTEM DIAGRAM. See Apendix 3.

Although superior to conventional flow charting in the representation of real-time assembly language algorithms, system diagrams cannot convey sequence timing information. To overcome this omission supplementary notes are required. Appendix 4 contains the supplementary notes required for the system diagrams presented in section 3.2.2.

3.2.2 Algorithms Investigated

The initial algorithm, EAMP00, detected the epoch amplitude parameter, A_I, on an epoch to epoch basis and normalised it to a preset level, L. The processing of the PCM samples, x(n), may be represented by equation (3.1).

$$x'(n) = \frac{L.x(n)}{A_I} \qquad (3.1)$$

The system diagram for this process is given in figure 3.2. Since all epochs were scaled to the same peak amplitude level, the background noise signal, as well as the speech signal, was "amplified" to the same level which enhanced the background noise. To obviate this, the manipulation of amplitude information for noise reduction was incorporated into the next algorithm, EAMP10. To reduce the background noise and yield speech of more acceptable quality to that of EAMP00, a peak amplitude threshold was introduced. If the peak amplitude of an epoch was less than the threshold level then all of

the samples forming the epoch were assigned a value of zero. The processing of the PCM samples, x(n), and the application of the threshold level may be represented by equation (3.2) which is given below:

$$x'(n) = \frac{L.x(n)}{A_I} \quad \text{if } A_I \text{ .GE. THRESHOLD}$$
$$= 0 \quad \text{otherwise} \qquad (3.2)$$

THRESHOLD was a variable stored within the 'receiver' and could be adjusted during program execution. This processing was identical to that of equaton (3.1) if A_I was greater than or equal to THRESHOLD. Figure 3.3 presents the system diagram for EAMP10.

The method of implementing equation (3.2) into a system would determine the transmitted amplitude information per epoch , I. If the tes-codewords (the epoch duration and shape descriptors) are transmitted, irrespective of whether the speech is active or inactive, no amplitude information is required - equation (3.1). However, if a unique one bit codeword is transmitted when the speech becomes "inactive" (A_I is less than THRESHOLD) and, on receiving the unique codeword the receiver synthesises all epochs with zero amplitude until the next unique codeword is received, the transmitted amplitude information per epoch, I, will equal 1/N where, N is the number of epochs synthesised with zero amplitude samples. In the extreme case (where alternate epoch peak amplitudes, A_I, are less than THRESHOLD) N will equal one and a maximum of one bit of amplitude information per epoch is required. Other implementations of equation (3.2) are possible and, in general, they will utilise diff-

erent quantities of transmitted amplitude information. However, the technique described above has set an upper limit of one bit of amplitude information per epoch and any implementation of equation (3.2) which requires more than one bit of amplitude information may be disregarded.

Since the previous process utilised one bit per epoch of amplitude information, this figure was regarded as a general 'yard stick' for further investigations.

The next algorithm to be developed, EAMP20, was directed towards the tracking of a generalised amplitude envelope of the signal rather than that of the individual epochs. To do this the amplitude parameter was detected over a group of N epochs, A_N, as well as on an epoch to epoch basis, A_I. Where N was set before program execution. Each epoch within the group was scaled such that the individual epoch amplitude parameters, the A_I's, were set equal to that of the group, A_N. Equation (3.3) defines the processing conducted.

$$x'(n) = \frac{A_N}{A_I}.x(n) \qquad (3.3)$$

The analogue-to-digital converter had a precision of eleven bits for magnitude. Since A_N was detected over a group of N epochs the transmitted amplitude information per epoch, I, was:

$$I = 11/N \quad \text{bits/epoch} \qquad (3.3(a))$$

From equation (3.3(a)) it was clear that as N increased, I decreased and when the processing was conducted over eleven epochs the amplitude information was once again one bit per epoch. However,

if N = 1 then the envelope factor degenerated into an individual epoch amplitude while at the other extreme, an infinite group size degenerates the system to that of EAMP00. The system diagram for this algorithm is given in figure 3.4.

Exclusion of the slow envelope variations may result in an increase in the number of bits required to represent the amplitude parameter on an epoch to epoch basis. However, a more efficient representation of amplitude might be achieved by applying a conventional speech coding technique such as Deltamodulation or Differential Pulse Code Modulation (see chapter 1) for the coding of the amplitude parameter and some variants of this were investigated.

The initial algorithm to be developed, EAMP30, which fell into this catagory employed Delta Modulation (see section 1.3) of the amplitude parameter. On an epoch to epoch basis the amplitude parameter, A_I, was detected and compared with its previous processed value, A'_{I-1}.

If	$\lvert A'_{I-1} \rvert$ .NE. 0		
and	A'_{I-1} .LE. A_I	then	$A'_I = A'_{I-1} + STEP$
or	A'_{I-1} .GT. A_I	then	$A'_I = A'_{I-1} - STEP$
If	$A'_{I-1} = 0$		
and	$\lvert A_I \rvert$ .LT. THRESHOLD	then	$A'_I = 0$
or	$\lvert A_I \rvert$ .GE. THRESHOLD	then	$\lvert A'_I \rvert = STEP$

STEP and THRESHOLD were variables stored within the receiver. THRESHOLD was implemented to prevent the magnitude of A_I from being 'decremented' to a negative value. All PCM samples within the

epoch, x(n), were then processed as defined by equation (3.4).

$$x'(n) = \frac{A'_I \cdot x(n)}{A_I} \qquad (3.4)$$

The values of STEP and THRESHOLD were adjustable during program execution. This enabled optimisation of the parameters for achieving, subjectively, the best possible quality and intelligibility. Figure 3.5 is the system diagram for this algorithm.

In an attempt to improve upon the quality and intelligibility of the speech produced by EAMP30, several Adaptive Deltamodulation algorithms were developed. The algorithms generally employed a recursive formula based upon the past history of the amplitude parameter. However, due to the computer word size, rounding errors and truncation errors constraints had to be imposed which severely limited the algorithms. The quality of speech produced by these algorithms was also subjectively appraised to be inferior to that of EAMP20 and EAMP30. The adaptive algorithms were therefore abandoned.

Progressing with the conventional speech coding techniques, an algorithm which utilised 2 bit Differential Pulse Code Modulation (DPCM) for signalling the epoch amplitude parameter was developed, EAMP50. On an epoch to epoch basis A_I was detected and compared with its previous processed value, A'_{I-1}.

If $|A'_{I-1}|$ .LE. $|A_I|$

then $A'_I = A'_{I-1} + STEPH$ }
} min[X_1, X_2]
then $A'_I = A'_{I-1} + STEPL$ }

where $X_1 = A_I - (A'_{I-1} + STEPL)$

and $X_2 = A_I - (A'_{I-1} + STEPH)$

If $|A'_{I-1}|$ .GT. $|A_I|$

then $A'_I = A'_{I-1} - STEPH$ }
} $\min[X_1, X_2]$
then $A'_I = A'_{I-1} - STEPL$ }

where $X_1 = A_I - (A'_{I-1} - STEPL)$

and $X_2 = A_I - (A'_{I-1} - STEPH)$

If $|A_{I-1}| = 0$

and $|A'_I|$ .GE. THRESHOLD then $A'_I = STEPL$

or $|A'_I|$ .LT. THRESHOLD then $A'_I = 0$

STEPL, STEPH and THRESHOLD were variables stored within the receiver which were adjustable during the program execution. This again enabled optimisation for achieving, subjectvely, the best possible quality and intelligibility. Each PCM sample, x(n), within the epochs was processed as defined by equation (3.4). Figure 3.6 is the system diagram for the algorithm EAMP50.

In section 3.3, which presents an informal subjective appraisal of the speech produced by the algorithms previously described, all algorithms are referred to by their codename i.e. EAMP00 etc. At this point it is informative to present a short summary of the algorithms developed, briefly stating their overall processing function.

EAMP00 : Scaling of the PCM samples such that all epoch peak amplitudes of the input signal are of the same magnitude.

EAMP10 : EAMP00 with peak amplitude threshold. If the peak

amplitude of an epoch was less than the threshold value the samples forming that epoch were set to zero.

EAMP20 : The scaling is performed over N epochs. In the data frame, the maximum value of the N peak amplitudes of the N epochs was detected and all N epochs were scaled to the same level as the maximum of the peak amplitudes. No thresholding employed.

EAMP30 : First of the algorithms to exploit a conventional speech coding technique. Deltamodulation of the epoch peak amplitude, on an epoch to epoch basis, was employed.

EAMP50 : This algorithm employed 2 bit Differential Pulse Code Modulation for signalling the epoch peak amplitude on an epoch to epoch basis.

3.3 Informal Subjective Appraisal

EAMP00 scaled the PCM samples within an epoch such that the relative magnitudes were unchanged, but all epochs had the same peak amplitude value. Low level signals which occurred during periods of inactive speech were therefore enhanced. The overall effect was to produce speech of very poor quality which had a harsh metallic characteristic normally associated with synthetic speech. The enhancement of inter-word noise gave the impression of low intelligibility. Listeners who were not familiar with the utterances had to concentrate before being able to understand what phrase was uttered.

The introduction of an amplitude threshold, EAMP10, for the

indication of active/inactive periods of speech produced a marked improvement in quality. A threshold level of approximately 36dB below the maximum peak magnitude of the speech waveform was found to produce the better quality. The threshold level, although established independently, coincided with that utilised by Al-Doubooni [39] in his TES coder.

Phillips and Thomas [57] investigated a level controller which, in some respects, was similar to the algorithm EAMP10. The controller was developed for implementation at the sending end of a noisy channel. However, the controller utilised two thresholds (see figure 3.7(a)) but the system reported was effectively a single threshold system (see figure 3.7(b)) because one of the thresholds had been set equal to the smallest quantum of the systems ADC.

The level controller differed to EAMP10 in its treatment of epochs with a peak amplitude less than the threshold level, T_1 in figure 3.7(b). In EAMP10, all signal samples of such an epoch were assigned zero amplitude. However, the level controller amplified the epochs samples by the appropriate gain given by the Input-Output characteristic of figure 3.7(b). In the two threshold system, if a peak amplitude was less than the second threshold, T_2 in figure 3.7(a), the epoch was not processed.

Although Phillips and Thomas reported that the algorithms produced speech which had "a significant improvement of intelligibility" and informal listening tests indicated that "the use of a double threshold might improve the quality still further" these conclusions were reached by comparing processed and unprocessed speech which had

been "transmitted" over a noisy channel. The speech produced by EAMP10 had not been subjected to any simulation of transmission. It is therefore not appropriate to compare the results of the work by Phillips and Thomas with those reported here.

However, with the algorithm EAMP10 the speech was still harsh with spasmodic clicking due to solitary epochs, within the silence period, having peak amplitudes greater than the threshold level. It was also discovered that some words were truncated causing the speech to sound disjointed. The truncation, whether at the beginning or at the end of an utterance, is a characteristic of employing a static threshold level for the suppression of background noise. Such thresholds introduce a sharp discontinuity between the two modes of operation of the algorithm, viz. speech/silence. There was no hysterisis in the decision process which could introduce a gradual transition from one mode of operation to the other. The truncation was a direct result of an unintelligent algorithm, it was incapable of discriminating between regions of speech signal and background noise. Therefore, when an utterance began (or ended) with a weak fricative, a weak plosive, had a nasal ending or was the trailing off of a vowel sound at the end of an utterance, then a degree of truncation of the utterance was experienced. The frequency of occurrence of this effect was therefore a function of the speech material. Although the truncation caused a loss of naturalness in the quality of the speech output, tests have not yet been conducted to determine whether it had any effect upon the intelligibility of the speech output.

The algorithm EAMP20 which involved processing groups of N

epochs was developed such that the quantity of amplitude information per epoch, I, was made variable as indicated by equation (3.3(a)). The value of N (the number of epochs over which the processing was conducted) was incremented over the range 1 to 11. This range of values were studied to examine the effect of decreasing the amplitude information per epoch from, the unprocessed state when N = 1 (in which case 11 bits of amplitude information per epoch are utilised) to the situation where N = 11 (in which case 1 bit of amplitude information per epoch are utilised). When N was varied from 1 to 2, this being equivalent to changing the amplitude information per epoch from 11 to 5.5 bits, a perceptible change in speech quality occurred. However subjectively, no deterioration in intelligibility was experienced.

As N varied from 2 to 4, the most obvious difference was the increased level of background noise for N = 4. However, it must be stressed that the background noise was not obtrusive. The change in level was noted because little change in speech quality had occurred. It was therefore difficult to distinguish between these algorithms.

The algorithm with N = 8 is of particular interest because King and Gosling [30] signalled the mean amplitude level of the speech waveform after every eighth TES codeword. In the algorithm studied here, with N = 8, the amplitude information per epoch was equivalent to 1.37 bits. The speech output was subjectively louder than the original and had a slight, although noticable, "tutting" during some utterances. This was attributed to the algorithm emphasising speech or aspirated sounds and was particularly noticable at the beginning of an utterance.

Because more epochs were captured in each 'frame', if a frame overlaped a fast transition from one speech sound to another or if there was a rapid build up in signal level, the smaller epochs within the frame become amplified. It was this which caused the over emphasis of the speech sound, especially when preceded by "silence". A segment of speech processed by EAMP20, with N = 8, is given in figure 3.8. The original bandlimited speech segment is given in figure 3.8(a) and the output of EAMP20 is given in Figure 3.8(b)

With N = 11, equivalent to one bit of amplitude information per epoch, similar effects to those described for N = 8 were heard. It was difficult, but not impossible, to discriminate between the algorithms when N equalled 8 and 11. As expected, for N = 11, the speech quality had suffered compared with the original yet it was considered to be superior to that produced by EAMP00 and EAMP10.

The speech ouput from EAMP20 (with N = 11) had a strained quality with a slight granular harsh sound. The speech was clearly intelligible and the inter-word distortions of EAMP00 and EAMP10 were noticeably absent thus the processed speech was made easier to listen to and more acceptable.

The first implemention which employed conventional speech coding techniques was EAMP30. This algorithm utilised Delta Modulation of the amplitude parameter on an epoch to epoch basis. The step size and threshold levels were adjustable during program execution and the optimum values were found to be approximately 34dB and 36dB below the maximum peak magnitude, respectively. These 'optimum' values were established during informal subjective appraisals of the

coders. Listeners were requested to adjust the step size and threshold level until the speech output was (in their opinion) the best quality achievable. Once the listeners were satisfied with the quality, the programme was halted and interrogated for the step size and threshold level.

The speech output by EAMP30 sounded strained and granular and for some utterances had an aspirated quality. The speech was intelligible and superior in quality to that produced by either EAMP00 or EAMP10. The inter-word noise was more noticeable for this technique than with EAMP20 but this characteristic was not detrimental to the overall speech quality. However, over long utterances some difficulty was experienced in differentiating between speech output by EAMP30 and EAMP20.

In an attempt further to improve the quality and intelligibility of the speech produced by EAMP30, several programmes were developed which utilised an adaptive step size. The adaption was, in general, based on either the previous value of the epoch amplitude parameter or a history of previous values. It was discovered that this approach gave no obvious improvement. The quality was generally inferior to that given by linear Delta Modulation, being coarse and 'clicky' or muffled and fuzzy depending on the adaption algorithm. The algorithms were also heavily constrained to prevent processor overflow. Subjectively, the intelligibility of the adaption algorithms were, in general, considered to be inferior to that of EAMP20 or EAMP30. Since the adaption algorithms were highly complex relative to EAMP20 or EAMP30 and unable to yield speech of similar or

better quality it was considered futile to persue them further.

The final algorithm implemented, EAMP50, again employed a conventional speech coding technique. EAMP50 coded the amplitude parameter on an epoch to epoch basis utilising 2 bit DPCM. With the threshold level set at 36dB below the maximum peak magnitude, the optimum speech output was achieved with the step sizes set at 42dB below the maximum peak magnitude and 20dB below the maximum peak magnitude.

The speech output was subjectively of better quality and less disturbing to listen to than that of either EAMP20 or EAMP30. Although the speech was of good quality and intelligible, some inter-word clicking was evident and during some utterances the speech had an aspirated quality.

3.4 Performance Assessments

During the informal subjective appraisals of the speech produced by the various algorithms it was realised that listeners experienced some difficulty distinguishing between EAMP20, EAMP30 and EAMP50. No single coding technique was thought to be of higher intelligibility than the others and a great deal of disagreement over preference occurred.

It was therefore proposed to conduct Diagnostic Rhyme Tests (DRT) to assess the intelligibility and Direct Comparison Tests (DCT) in an attempt to establish which, if any, was the more preferable. In appendix 7 an outline of the DRT and DCT is given.

In both tests the speech output from the microcomputer was bandpass limited (0.3 to 3.4kHz) using an eighth order Barr and Stroud variable filter EF3 and recorded using a JVC cassette recorder (CD 1635 Mk II) onto BASF C60 LH SM cassettes.

3.4.1 Diagnostic Rhyme Test (DRT).

These tests were undertaken in order to determine the differences, in terms of intelligibility, of the synthesised speech produced by:

(i) Grouped Epochs (N = 11), EAMP20

(ii) Delta Modulation, EAMP30

(iii) 2 bit DPCM, EAMP50

in comparison with

(iv) 11 bit/sample PCM control

The Diagnostic Rhyme Tests (DRT's) were conducted with a group of 10 listeners. The tests of the three coding techniques and the PCM control were conducted in one session lasting approximately 40 minutes.

Table 3.1 presents the Chance Adjusted Percent Correct (CAPC) scores produced by the DRT's for each listener in each test. Included in the table are the average CAPC scores for each test presented. Table 3.2 presents the average CAPC scores for each perceptual phonemic attribute in each test. The results given in Table 3.2 are also presented graphically in figures 3.9 and 3.10.

Before examining the results achieved by each of the coding

techniques it is instructive to examine the scores of the control test. From Table 3.1 it is observed that the CAPC scores for the control test varied over a small range of values (-3.3 to +2.1%) about an average value of 95.1%. Table 3.2 highlights that the phonemic attribute which was most difficult to distinguish was Graveness, which had an average CAPC score of only 80%. Voicing also proved to be a problem in the control test, achieving a score of 95%. The remaining phonemic attributes attained scores greater than 97%.

Inspection of Table 3.1 revealed that the individual subjects did not achieve CAPC scores for EAMP20, EAMP30 or EAMP50 which were greater than that of the control test. However, one listener (No.2) scored the same for EAMP50 as for the control test. It can also be seen that there is a small difference of 0.4% in the average CAPC scores for EAMP20 and EAMP50. Such a small difference is negligible because the average values presented were calculated from the 10 individual subject scores for each test. The individual scores had been rounded to one decimal place and so had the average scores. It would therefore be incorrect to claim that EAMP20 is more intelligible than EAMP50 based upon the scores achieved. However, it is quite obvious that these algorithms yielded a greater intelligibility than EAMP30. All listeners, except for No. 9, achieved CAPC scores for EAMP20 and EAMP50 which were greater than that of EAMP30.

Table 3.1 shows that, on average, EAMP20 and EAMP50 have similar intelligibility yet individual listeners preferences varied considerably. In order to gain some insight in to the differences

of the various algorithms the scores for each perceptual phonemic attribute of each test must be considered. These are presented in Table 3.2 and figures 3.9 and 3.10. Examination of the scores for each phonemic attribute reveal that:

(i) Nasality : This attribute was impervious to distortions caused by the coding techniques and no system yielded a score less than 98.8%.

(ii) Voicing : This attribute EAMP50 scored effectively the same as the control test. However, EAMP30 gave a significant loss of this attribute scoring only 81.7%.

(iii) Compactness : Again EAMP30 was the process which yielded a poor representation of this attribute, scoring only 87.5%. EAMP20 and EAMP50 achieved scores greater than 95%.

(iv) Graveness : This attribute only achieved a score of 80.6% for the control test. EAMP50 attained a similar score and EAMP30 was again the technique to score the least (69.4%).

(v) Sibilation : Although the control test achieved 97.5% for this attribute all three techniques produced poor representations of this attribute. The highest score achieved was 72.5% by EAMP20 while EAMP50 produced the least at 56.3%.

(vi) Sustention : The discrimination of this attribute in EAMP30 was very poor (68.7%) in comparison with EAMP20 and EAMP50 which achieved greater than 92%.

From the results presented so far it can be seen that to the majority of the listeners EAMP30 was thought to have inferior intelligibility to that of either EAMP20 or EAMP50. However, the choice of "which, if either, of EAMP20 and EAMP50 was of higher intelligibility?" varied considerably amongst the listening panel and yet they achieved similar average CAPC scores. From inspection of Table 3.1 we observe that five listeners found EAMP20 more intelligible than EAMP50, four listeners found EAMP50 more intelligible than EAMP20 and one listener found EAMP20 and EAMP50 of equal intelligibility.

The CAPC scores for each attribute indicated that EAMP20's scores were at least 2% less than those of EAMP50, except for nasality which achieved an equal score and sibilation for which EAMP20 scored 15% better than EAMP50. Relative to the scores achieved from the control test, with the exception of sibilation, EAMP50 achieved almost 100% for all attributes. It was the sibilation which caused the mean CAPC scores to be very similar.

As stated previously, the results of Table 3.1 indicated that EAMP20 was capable of producing speech of greater intelligibility to that of EAMP50, albeit with only 0.4% difference. On the other hand, inspection of Table 3.2 reveals that it was the single phonemic attribute of sibilation which caused a reduction in EAMP50's average CAPC score. It was therefore decided to conduct preference tests in order to determine over which of the systems, if either, listeners would prefer to receive speech transmissions. For this purpose Direct Comparison Tests were conducted.

3.4.2 Direct Comparison Tests (DCT)

The DCT's were conducted employing the group of 10 listeners who took part in the DRT's discussed in the previous section. The tests were conducted in individual sessions lasting approximately 10 minutes.

The results of these tests are summarised in the preference matrix of Table 3.3. The results indicate that EAMP20 and EAMP50 are generally more preferable to EAMP30, the preference between EAMP50 and EAMP30 being very clearly indicated in these tests with all listeners indicating this distinction. However, the distinction between EAMP20 and EAMP30 was not so precise. During the first presentation one listener found no preference and another preferred EAMP30 to EAMP50. On the second presentation, again one listener found no preference but the remainder preferred EAMP20 to EAMP30.

Inspection of the preference scores between EAMP20 and EAMP50 demonstrate a very slight bias to EAMP50. However, from such a small sample no clear preference between EAMP20 and EAMP50 has emerged.

3.5 Conclusions

The results of the investigations reported indicated that amplitude information of the order of one bit per epoch produced acceptable quality in the processed speech. The Diagnostic Rhyme Tests (DRT) yielded very similar scores of the order of 88% intell-

igibility for algorithms which inserted one amplitude value for all epochs in a group (EAMP20 was set to one 11 bit amplitude codeword for 11 epochs) and for algorithms which signalled the difference in amplitude for successive epochs employing Differential Pulse Code Modulation (EAMP50 - 2 bit DPCM). The DRT scores in both cases were appreciably less than for the unprocessed control PCM samples.

The Direct Comparison Tests (DCT) indicated that EAMP20 and EAMP50 were generally more preferable to EAMP30, with a very clear preference being exhibited for EAMP50 compared with EAMP30. The preference scores for EAMP20 and EAMP50 produced a very slight bias to EAMP50, but due to the small sample employed no firm conclusions may be drawn from this result.

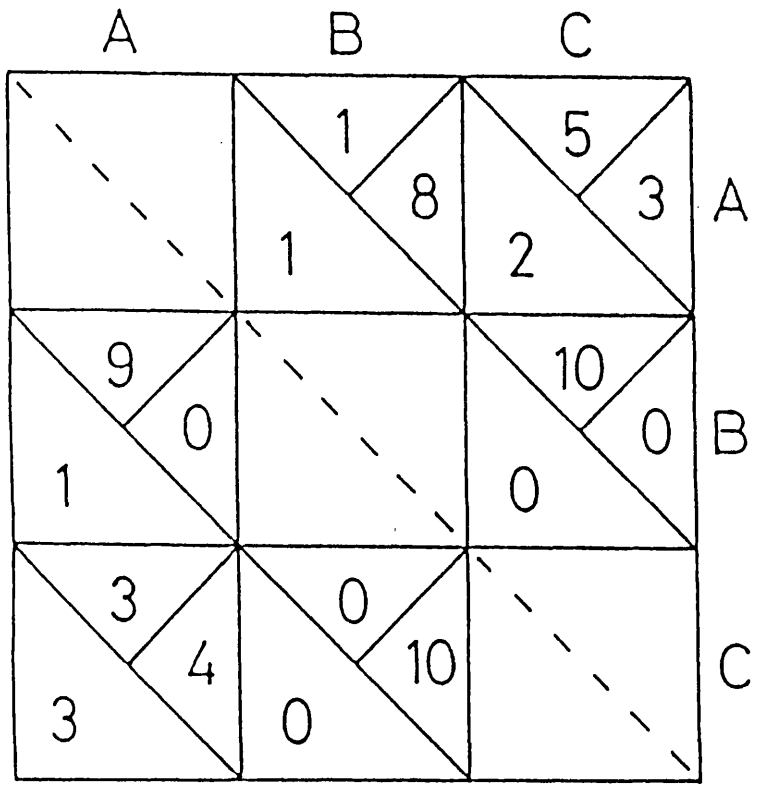
It must therefore be concluded that even though a minimum of one bit per epoch has been indicated as sufficient for conveying amplitude information, the true intelligibility from a TES system will be dependent upon the amplitude encoding/decoding algorithm as well as a number of other features of the TES coder/decoder.

	Test No.				
	1	2	3	4	
ATTRIBUTE	EAMP20	EAMP30	EAMP50	CONTROL	MEAN
Voicing	92.3	81.7	94.8	95.1	91.0
Nasality	100.	98.8	100.	100.	99.7
Sustention	93.7	68.7	96.3	98.1	89.2
Sibilation	72.5	67.5	56.3	97.5	73.4
Graveness	74.4	69.4	80.0	80.6	76.1
Compactness	96.9	87.5	100.	99.4	95.6

Table 3.2 : Average Chance Adjusted Percent Correct scores of each phonemic attribute for each test.

	Test No.			
	1	2	3	4
Subject No.	EAMP20	EAMP30	EAMP50	Control
1	88.0	71.7	88.0	95.6
2	84.8	69.6	91.3	91.3
3	86.9	66.3	88.0	94.6
4	90.2	78.3	89.1	93.5
5	86.9	76.1	83.7	93.5
6	92.4	80.4	90.2	95.6
7	88.0	78.3	89.1	94.6
8	86.9	82.6	90.2	94.6
9	88.0	91.3	86.9	96.7
10	91.3	72.8	82.6	95.6
Average DRT Score	88.3	76.7	87.9	94.6

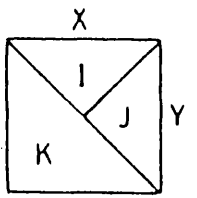
Table 3.1 : Chance Adjusted Percent Correct (CAPC) score for each listener in each test and the Average CAPC score for each test.



A : EAMP20

B : EAMP30

C : EAMP50



I - Number of listeners prefering X to Y.

J - Number of listeners prefering Y to X.

K - Number of listeners not expressing a preference.

Table 3.3 : The preference matrix for the three amplitude signalling strategies derived from the Direct Comparison Tests. The values above the diagonal shows the preference in the 1st hearing and the value below shows the preference in the 2nd hearing.

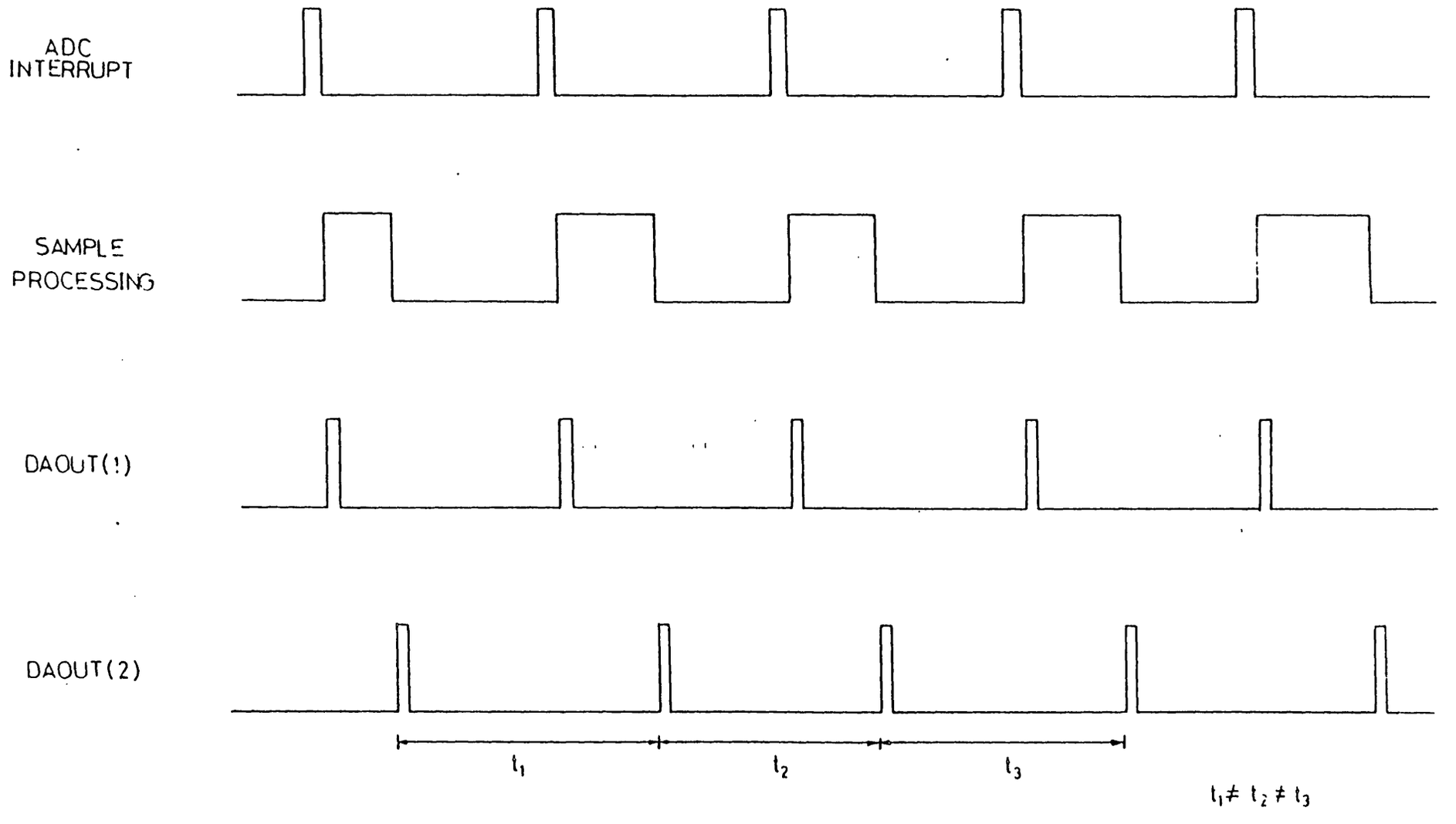


Figure 3.1 : Representation of sample processing and data output timing.

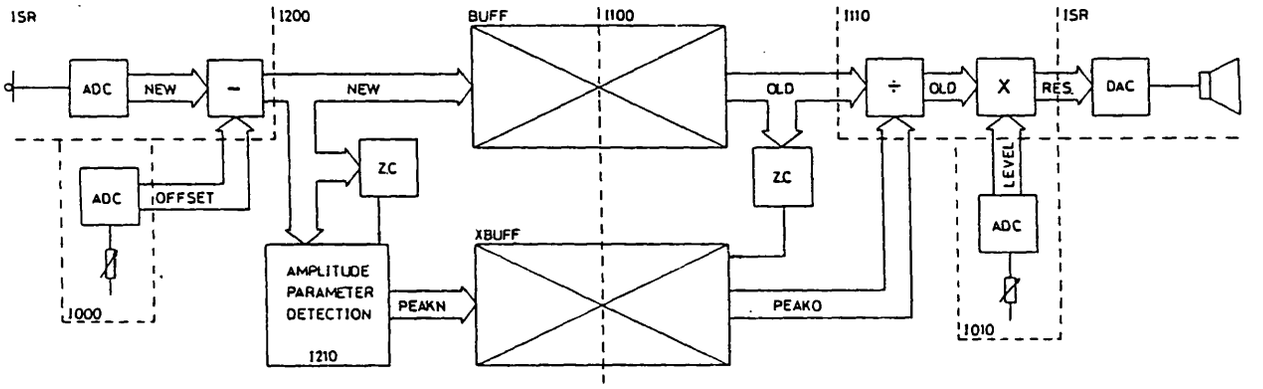


Figure 3.2 : System diagram for the process EAMP00.

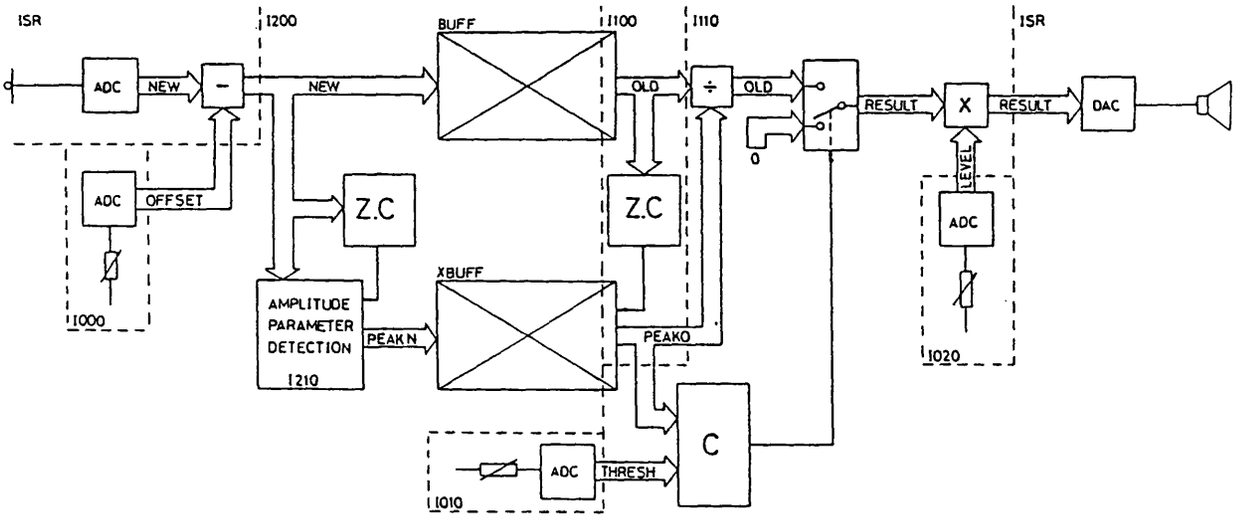


Figure 3.3 : System diagram for the process EAMP10.

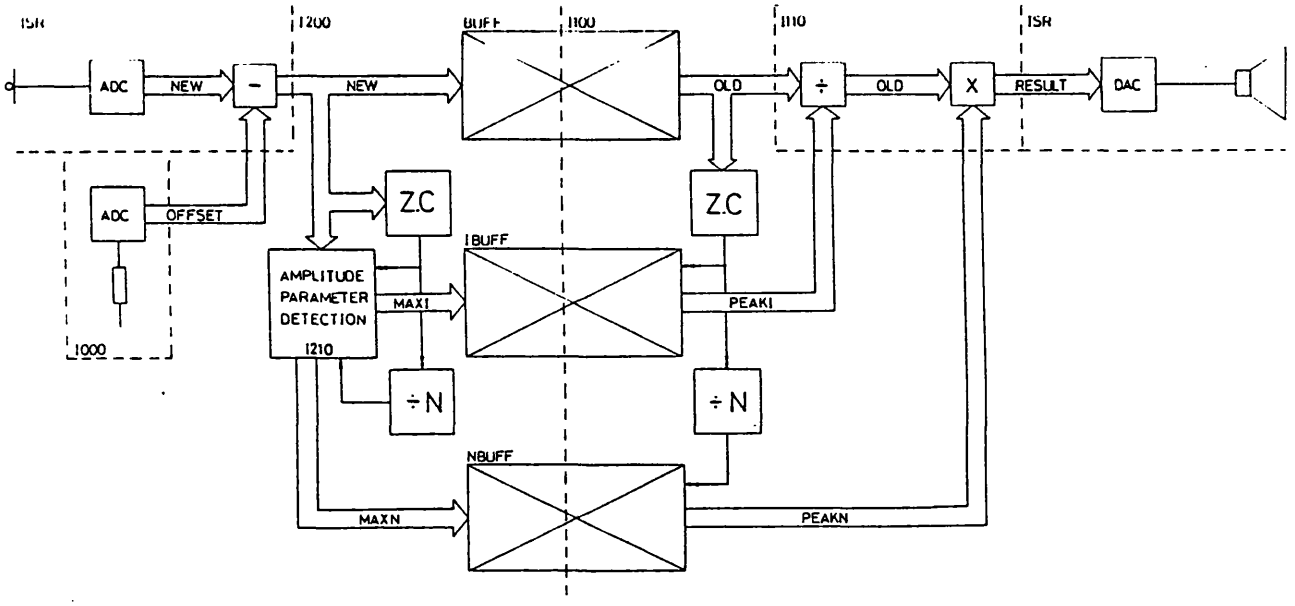


Figure 3.4 : System diagram for the process EAMP20.

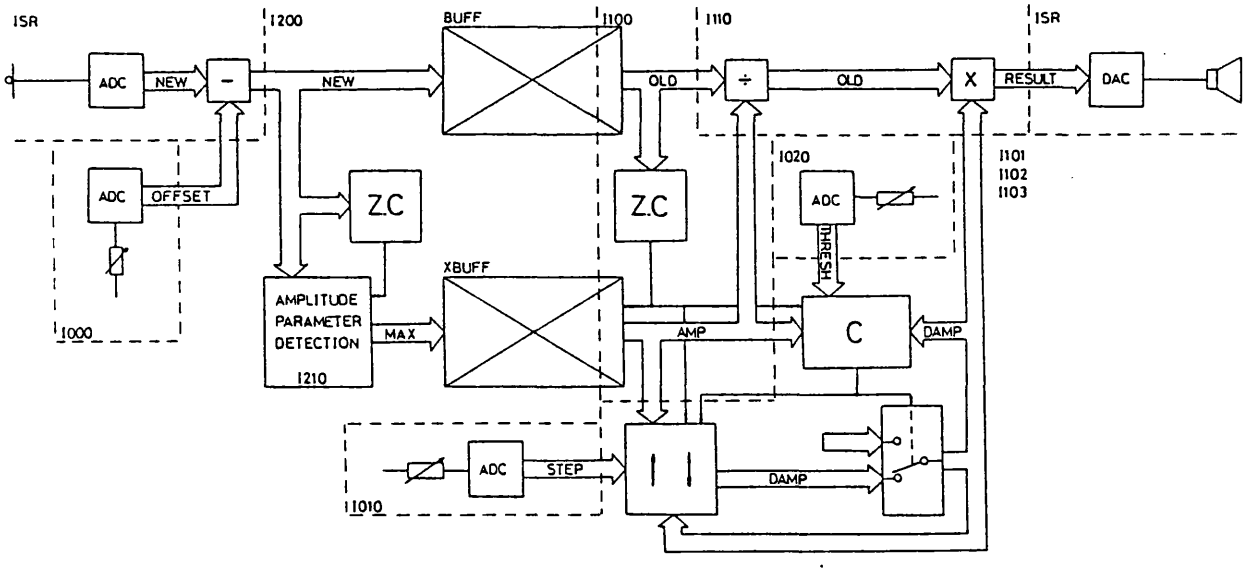


Figure 3.5 : System diagram for the process EAMP30.

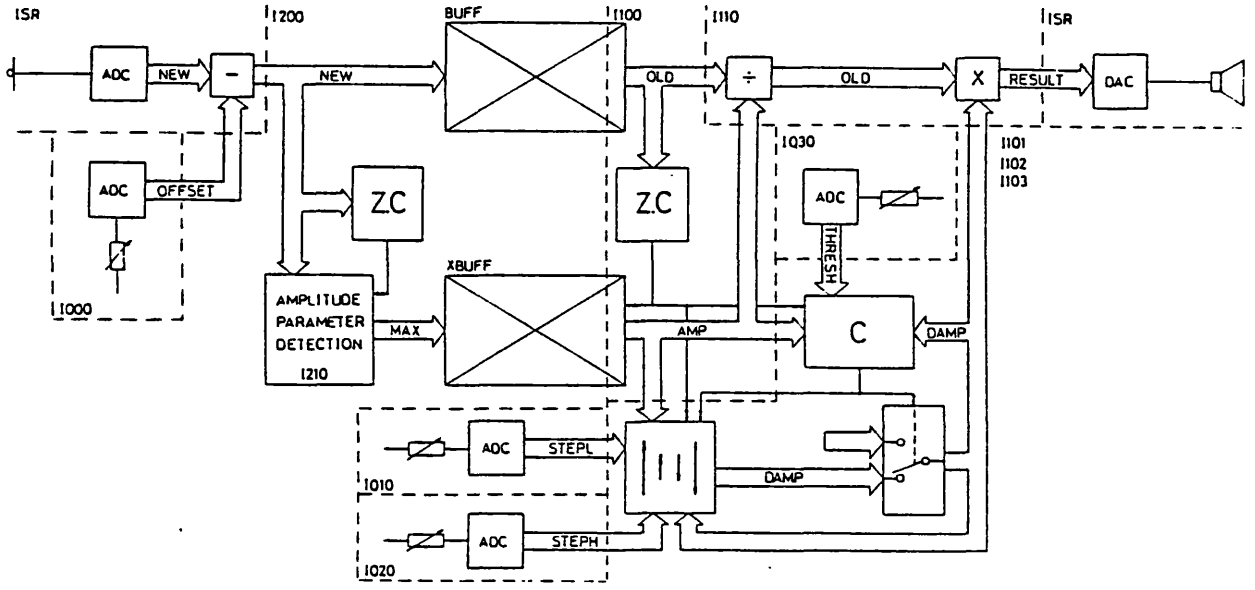


Figure 3.6 : System diagram for the process EAMP50.

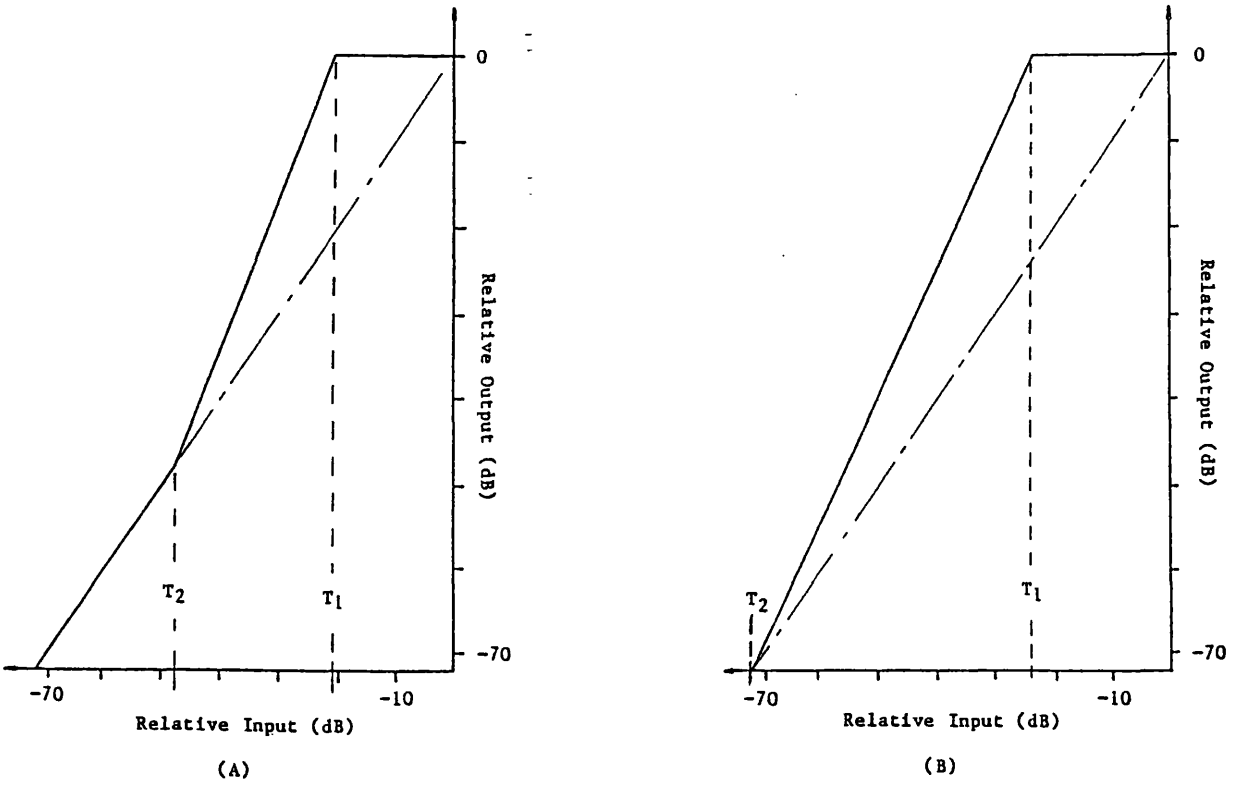


Figure 3.7 : Input-Output Characteristics of a Feed-Forward Level Controller[57]
(a) with T_1 = -21db and T_2 = -48db and (b) with T_1 = -24.1db and T_2 = -72dB
(the smallest system quantum) effectively forming a single threshold system.

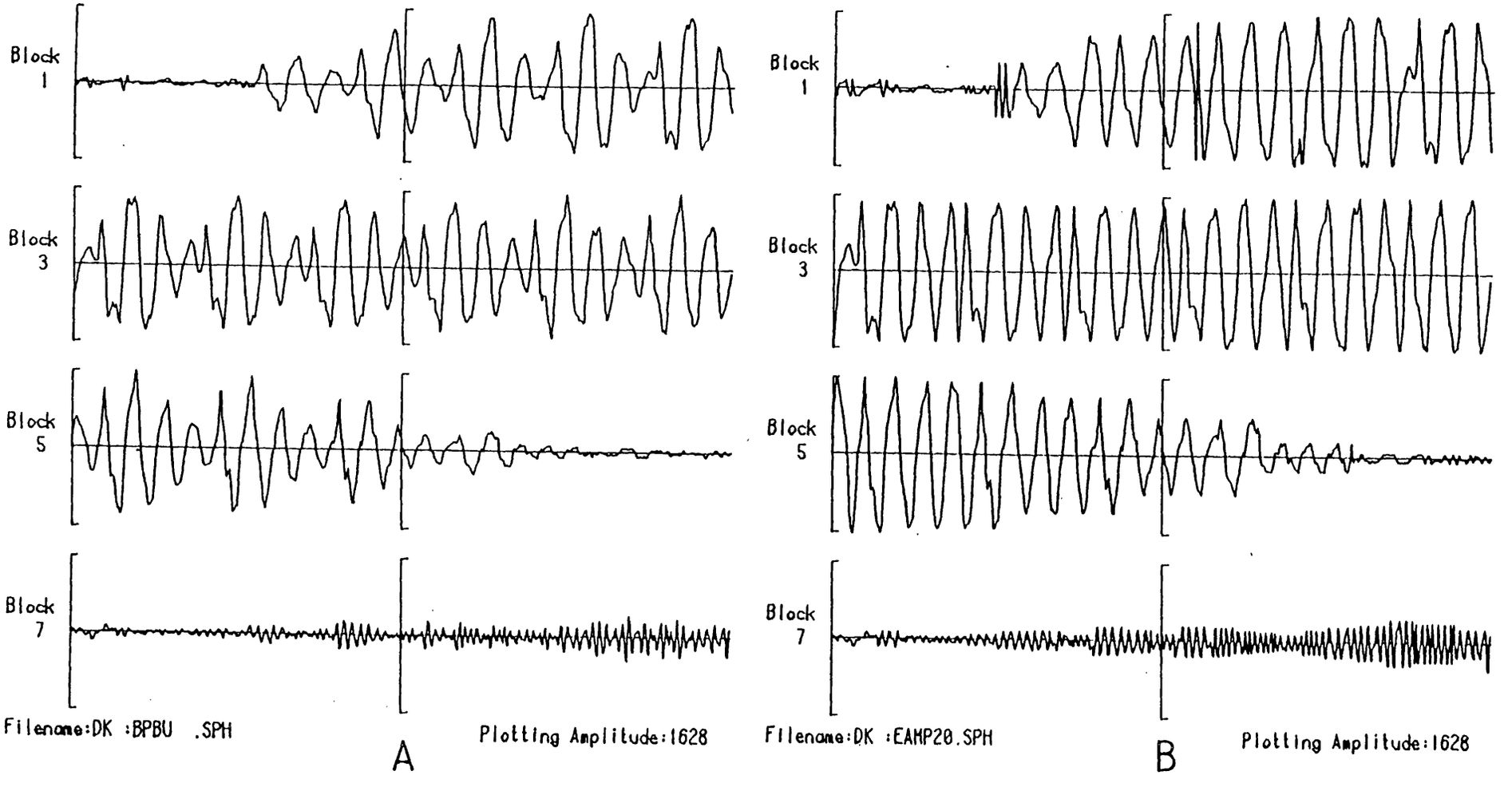


Figure 3.8 : (a) Segment of the original band-limited speech.

(b) After processing by EAMP20 with N = 8.

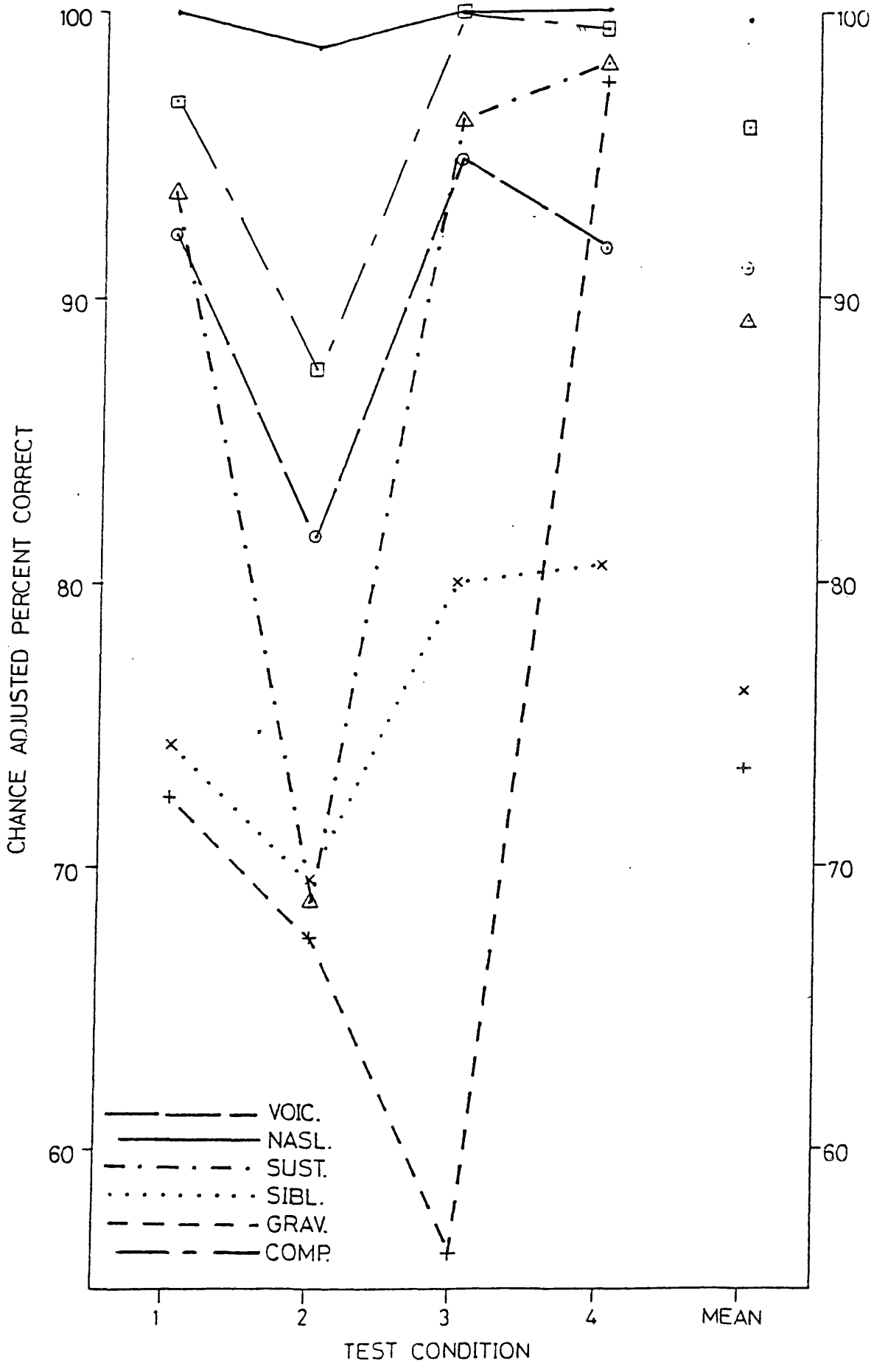


Figure 3.9 : Diagnostic profile for four speech processing techniques.

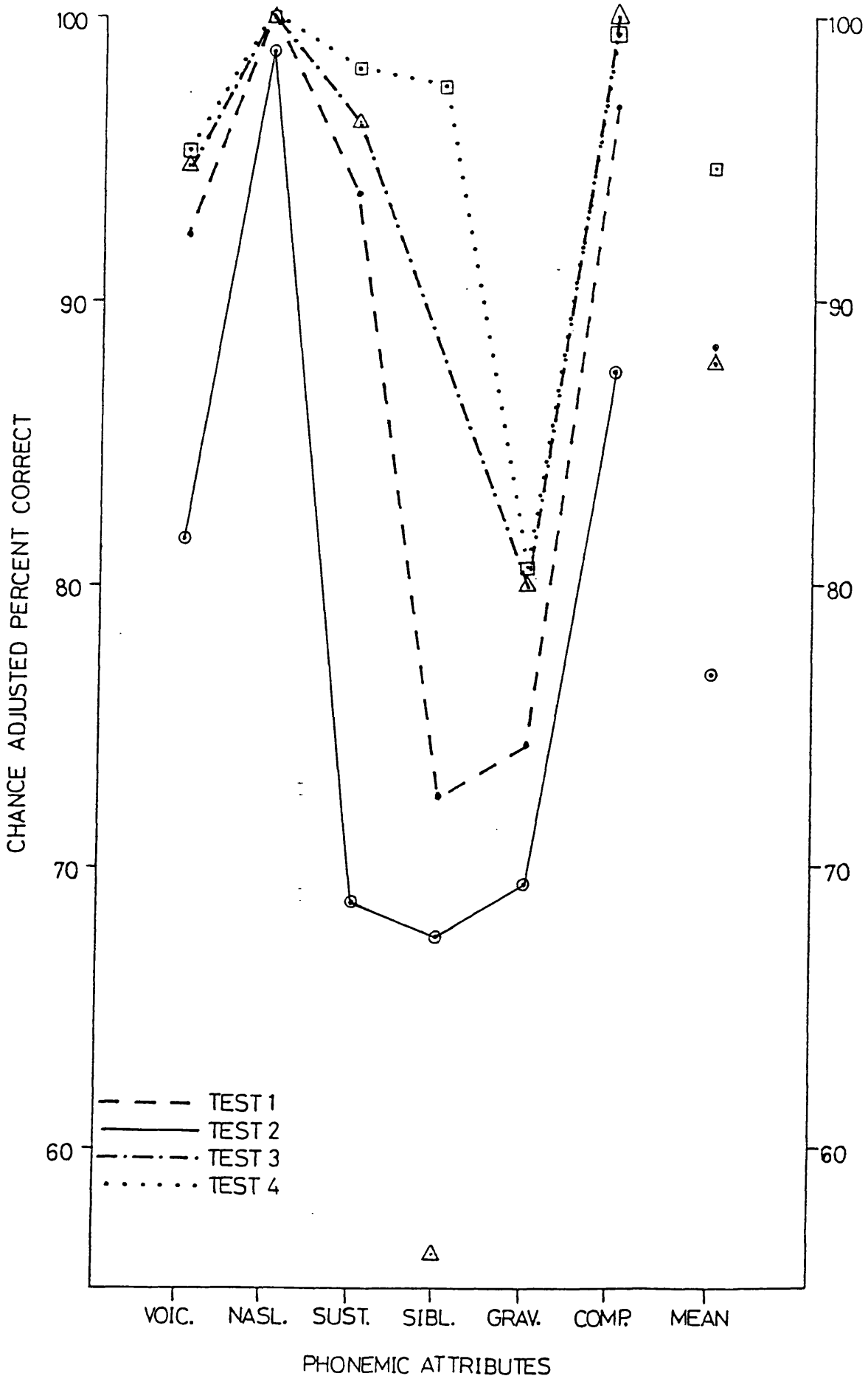


Figure 3.10 : Diagnostic profile for four speech processing techniques.

Chapter 4

Median Filtering

4.1 Introduction

Linear time invariant filters, or equivalently linear smoothers have recieved considerable attention in digital signal processing theory and application [59,60]. The frequency selective filtering of linear systems enables the power spectrum at the output of the filter to be determined when the input is a stationary random process with known power spectrum. As a consequence of this it is possible to eliminate unwanted signal components (like additive noise), if the signal processing is to be based exclusively on the frequency content of the processed signal and the contaminating signal. However, for some applications, linear filtering is inadequate due to the nature of the data to be smoothed. If the signal is a composite of the sum of a high frequency noise component and information bearing sharp edges, then a linear smoother, upon removing the noise signal would also smear out the edges of the original signal. Such smearing of the data is unacceptable in many applications. To overcome this difficulty, the use of non-linear smoothing devices must be contemplated. Tukey [61] is generally credited with the idea of introducing non-linear filters based on moving sample medians of the input signal. These are referred to as Median Filters.

The signals processed using Median and Moving Average filtering techniques are not unlike those produced by the sequential plots.

Figures 4.1(a),(b),(c) and (d) are sequential plots of epoch durations extracted from a coded speech file. Inspection of these plots reveals a number of features due to the characteristics of

speech. The most prominant feature being the periodicity within the epoch sequences due to strong voiced sounds. The epoch sequences have a definite structure within each 'period', plus locally small variations (in comparison with neighbouring periods) of the order of 0.02ms. Such periodicity can be observed in figure 4.1(a), samples 0 - 50 and 110 - 200.

The epoch sequences due to voiced sounds which are not so strongly pronounced also exhibit periodicity over very short intervals. A number of examples of this feature are observed in figure 4.1(b), samples 10 - 30 and 215 - 235 and figure 4.1(c), samples 200 - 215.

Within the epoch sequences, unvoiced sounds manifest as short epoch durations (.LT. 0.6ms) and the epoch sequences are very erratic within this range. However, occasionally during unvoiced sounds, epoch durations greater than 0.6ms do occur in isolation producing an impluse-like appearance in the epoch sequence. Examples of this may be seen in figure 4.1(d), samples 150 - 250.

Some epoch sequences appear to have little or no structure. The epoch durations vary between 0.25 and 1.25ms and variations which are small in comparison with the local topography also exist. Such sequences generally occur at the transition boundary between voiced and unvoiced speech, stop consonants and silence. This is also true for the inverse case. Examples of such sequences occur in figures 4.1(a), samples 75 - 105, 4.1(c), samples 0 - 120 and 4.1(d), samples 0 - 130.

As described in Chapter 2 Section 2.2, the epoch durations corresponding to the silence intervals between utterances had been removed. However, periods of large epoch duration (.GT. 1.5ms) still existed. Such segments were believed to be due to the intonation of the speech at the end of one utterance and the beginning of the next. In figure 4.1(b) the epoch durations appear to oscillate about the 1.75 ms level. It was surmised that this feature was a result of a 'tone-like' segment of speech with a gradually decreasing d.c offset.

When developing and/or implementing data reduction algorithms, features such as the locally small variations (observed within strong voiced sounds), the impulse-like structure during voiced sounds or the oscillatory effect of the epoch lengths, cause a significant increase in either coder complexity and/or data required for the adequate representation of the sequence.

Very little is known concerning the perceptual importance of these features and what the effect upon the overall synthesised speech quality would be if these features were either attenuated or eliminated. It was hypothesised that if a parameter pre-processing algorithm were developed which attenuated or eliminated these features previously described, whilst retaining the basic structure of the sequence and without causing significant degradation in quality of the synthesised speech, coding of the pre-processed epoch sequence for data reduction could be implemented with greater efficiency and/or less complexity.

The simulation algorithms developed smoothed the epoch duration sequence only. The amplitude and extrema information was not pro-

cessed. The speech was synthesised employing the algorithm developed by Al-Doubooni (see chapter 2)

Section 4.2 describes the linear, non-linear and dual stage smoothers investigated for the pre-processing of the epoch sequences and the results achieved using these algorithm are presented in section 4.3. Section 4.4 presents the conclusions of this investigation.

4.2 Smoothing Algorithms

The algorithms investigated as possible candidates for attenuating, if not eliminating, many of the features previously described consisted of two linear, two non-linear and four dual stage smoothers. In the following sections, 4.2.1 to 4.2.3, the algorithms implemented are outlined.

4.2.1 Non-Linear Smoothing

Median filtering is a non-linear signal processing technique useful for noise suppression [62]. Median filters have several interesting characteristics which can, in some applications, make them superior to linear filters. If a signal has impulse-like components, median filtering can eliminate them without significantly modifying other components and, if a signal contains step-like components, median filtering can preserve these discontinuities. Important applications in the processing of signals, where edges carry information have been reported in the literature. Rabiner et al [63]

used median filters to extract the "smooth" component of a speech signal and combined median filters with other stages of linear systems to smooth noisy sequences of a discontinuous signal. Jayant [64] proposed a median based smoother to improve digital speech quality in the presence of channel errors. Finally, Huang et al [65] and Frieden [66] have used median filters in image processing applications.

However, the majority of existing results are based upon empirical work and few theoretical aspects of median filtering have been published. Recently, Gallagher and Wise [67] studied and defined various concepts associated with median filters. In particular they defined the following signal characteristics.

A 'constant neighbourhood' is a region of at least N+1 consecutive points, all of which are identically valued.

An 'edge' is a monotonically rising or falling set of points surrounded on both side by constant neighbours.

An 'impulse' is a set of N or less points whose values are different from the surrounding regions and whose surrounding regions are identically valued constant neighbours.

4.2.1.1 One Dimensional Median Filtering

Median filtering is a discrete time process in which a window that spans 2N+1 signal samples, where N is the order of the median filter, is stepped across the input signal. At each step, the points

inside the window are ranked according to their values, and the median value (middle number in size) of the ranked set is taken to be the output value of the filter for each window position. An example is given below for N = 1 and 2.

Input	1	4	3	3	6	4	8	7	8	9	10
Output (N=1)	x	3	3	3	4	6	7	8	8	9	x
Output (N=2)	x	x	3	4	4	6	7	8	8	x	x

4.2.1.2 Initial and Final Conditions

In order to implement a median filter the algorithm must account for start-up and end effects at the end points of the sampled signal. In the example given above the initial and final conditions were not calculated and the missing samples were replaced by x. Rabiner et al [63] investigated several techniques for generating a set of initial and final values, including constant, linear and quadratic extrapolation. For most applications, constant extrapolation from the initial and final data points proved to be entirely adequate. For other methods of treating the start-up problem where less emphasis is given to the first and last values encountered see reference [61].

Unlike many applications, the initial and final conditions are not so critical for the epoch sequences because these epochs, when synthesised, correspond to the beginning and end of an utterance which is generally a low level signal. Therefore, the simulation algorithm employed constant extrapolation of the initial and final epoch durations.

The algorithm developed for the simulation of Median filtering required the user to specify the order of the median filter to be simulated (maximum order of 5) and the input and output data files. A flow chart of this algorithm is given in figure 4.2(a). Investigations were limited to the simulation of first and second order median filters because the initial results indicated that higher order filters would give little further benefit.

4.2.2 Linear Smoothing

As stated earlier in section 4.1, linear smoothers (time invariant filters) are useful for eliminating unwanted signal components based on frequency content. The linear smooothers employed by Rabiner et al [63] were 19 point Finite Impulse Response (FIR) low-pass filters. However, the epoch sequences cannot be described in terms of frequency components because of the variable rate at which they are generated and the application of a smoother similar to that employed by Rabiner et al would result in excessive distortion of the epoch sequence.

A technique for reducing the variations within the epoch sequence involving averaging adjacent samples, is termed Moving Average Filtering. If the number of adjacent samples in the averaging is kept small, then slowly varying components are retained while the impulse-like components are attenuated. A moving average filter of order N-1, where N is the number of samples averaged, is defined by:

$$Y(n) = \sum_{m=0}^{N-1} A_m \, x(n-m) \qquad (4.1)$$

where all the coefficients, A_m, are equal to 1/N. Equation (4.1) is also the difference equation of an N point FIR low-pass filter.

In order to implement a moving average filter, once again, initial and final conditions must be taken into account. For reasons previously explained in section 4.2.1.2, constant extrapolation was utilised for the generation of a set of requisite initial and final values.

The algorithm developed for the simulation of a Moving Average filter required the user to specify the order of the moving average filter to be simulated (maximum order of 10) and the input and output data files. A flow chart of the algorithm is presented in figure 4.2(b). As in the case of the median filters these investigations were limited to simulation of first and second order filters only.

4.2.3 Dual Stage Smooothing

Linear smoothers separate signals based on their (approximately) non overlapping frequency content. Signal separation using non-linear smoothers is best performed by considering whether the signal content can be termed rough (noise-like) or smooth [68]. Thus a signal x(n) can be regarded as

$$x(n) = S[x(n)] + R[x(n)] \qquad (4.2)$$

Where R[m] is the rough part of the signal and S[m] is the smooth part of the signal. Figure 4.4 shows a block diagram of a simple smoothing algorithm, the output of which is an approximation

to S[x(n)]. Since the approximation in many instance is not adequate a second stage of smoothing is incorporated into the smoothing algorithm as shown in figure 4.5.

Since $$y(n) = S[x(n)] \tag{4.3}$$

then $$v(n) = x(n) - y(n) = R[x(n)] \tag{4.4}$$

The second stage smoothing of v(n) yields a correction signal which is the 'smooth of the rough' and is added to y(n) to give z(n), a refined approximation to S[x(n)]. The signal z(n) satisfies the relation:

$$z(n) = S[x(n)] + S[R[x(n)]] \tag{4.5}$$

If v(n) = R[x(n)] exactly ie. the smoother were ideal, then S[R[x(n)]] would be identically zero and the correction term would be unnecessary.

In order to implement the smoothing algorithm of figure 4.5 as a realisable system account must be taken of the delays in each path of the smoother. Figure 4.6 shows the block diagram of a realisable version of the smoother of figure 4.5.

To simulate the dual stage smoother an algorithm was developed which:

- accounted for the delay introduced by each stage of smoothing.
- summed or differenced the input signals.

The user specified the operation required (summation or difference), the two input files and the output file. A flow chart of the

algorithm is given in figure 4.3

Simulating the dual stage smoothers using the 'black box' principle, enabled combinations of smoothers to be implemented provided the correct delays were incorporated within the parallel branches of figure 4.6. The algorithms investigated are listed below in Table 4.1.

First Stage	Second Stage	Abrv.
First order Median	Second order Median	DSA
First order Median	First order Moving Average	DSB
First order Median	First order Median	DSC
Second order Median	First order Median	DSD

Table 4.1 Dual Stage Smoothing Algorithms Investigated

As well as the algorithms listed in Table 4.1, a first and second order median filter were cascaded and implemented as the first stage of a dual stage smoother with a second stage comprising of either:

(a) First order Median

(b) Second order Median

(c) Cascade of (a) and (b)

The output of the algorithms was highly distorted and the speech, synthesised from the resulting epoch sequences, was totally unintelligible. Very little information could be gained from inspection of these sequences and they have therefore been omitted.

4.3 Results

The investigations were conducted in two stages. Initially, only the first and second order moving average and median smoothing algorithms were simulated.

Comparisons of the input and output sequences indicated that higher order moving average smoothers were required to produce adequate smoothing. However, a further increase in the extent to which the sequences were smoothed would have resulted in highly distorted synthesised speech. Also, to implement the moving average filter as the first stage of a dual stage smoother would have little effect because: the difference signal, v(n), would consist of small variations about zero. Thus w(n), the smoothed signal derived from v(n), would generally be zero and therefore not introduce any correction. Thus the second phase of these investigations simulated the dual stage smoothers listed in Table 4.1.

For the purpose of making comparisons, the epoch sequences within the output files corresponding to figures 4.1(a) and (c) were isolated and plotted. Several sections of the isolated epoch sequences were selected and employed for speech synthesised. Using a Digital Fast Fourier Transformation software package the power spectrums for the segments of synthesised speech were calculated and plotted.

Section 4.3.1 gives a comparison/discussion of the epoch sequences while in section 4.3.2 the quality of the synthesised speech is discussed. Finally, in section 4.3.3 the power spectrums are

discussed Table 4.2 provides a cross reference and guide to the relationship between the diagrams referred to in sections 4.3.1, 4.3.2 and 4.3.3.

4.3.1 Epoch Duration Sequence Comparisons

Figures 4.1(a) and 4.1(c) present the epoch duration sequencies input to the smoothing algorithms.

The First Order Moving Average (FOMA) smoother attenuated a significant proportion of the impulse-like features present in the epoch sequences (figure 4.7(b)), and eliminated the locally small variations observed in the periodic segments. However, some epochs within the periodic sections exhibited substantial distortions which resulted in epoch extentions of up to 75%. These epochs corresponded to a segment of strong voiced sound.

The Second Order Moving Average (SOMA) smoother produced greater attenuation and elimination of the impulse-like features (figure 4.8(b)) but resulted in 100% epoch extention during strong voiced sounds. However, a sequence of samples (No. 0 to 50 of figure 4.8 (a)) was not severely distorted and retained all but its minor features.

The first non-linear smoother implemented was a First Order Median (FOM) smoother. This smoother had a similar effect on the epoch sequences to that produced by the FOMA, resulting in the elimination of the impulse-like features and locally small extrema (figure 4.9(b)). Comparing the output and the periodic input sequence

(figures 4.1(c) and 4.9(a)) demonstrates only too clearly the effectiveness of a median smoother. The initial periodic sequence retained all but the minor features while the epochs corresponding to the strong voiced sound were distorted and in some instances, experienced extensive epoch extention.

The Second Order Median (SOM) smoother resulted in almost total elimination of the periodic sequence as well as the impulse-like features. The resulting sequences contained little of the desired features and served only to highlight the very general trend of the original sequence (figures 10(a),(b)). Obviously, a SOM smoother alone was too severe for the processing of epoch sequences.

The first dual stage smoother (DSA) to be implemented produced an output very similar to that of a FOM, with less attenuation of the impulse-like features. The similarity between DS1 and FOM is believed to be due to the nature of the difference signal, v(n). Since the ouput of the FOM was similar to the original a large proportion of the difference signal, v(n), was zero and the output signal of the SOM, w(n), with v(n) as the input signal was therefore also generally zero. Thus the output of the dual stage smoother, z(n), of figure 4.6, was approximately equivalent to the ouput of the FOM.

In order to combat these effects the second dual stage smoother (DSB) employed a FOMA as the second stage of smoothing while the first stage remained a FOM. The resulting output epoch sequences retained the majority of their periodic structure and only the first two periods (figure 4.12(a) samples 110 - 120) experienced substantial epoch extention. The remainder experienced relatively minor

distortions (0.1 or 0.15ms). The impulse-like features remained present to a greater extent but the smoother still eliminated the larger impulses, figure 4.12(b).

The third dual stage smoothing algorithm to be implemented (DSC) utilised FOM smoothers in each stage. The reasons for investigating this configuration were the same as those of the previous algorithm, namely: Where a SOM produced a signal, w(n), which was mainly zero in DSA, it was hypothesised that a FOM smoother may produce a correction signal which was non-zero for a greater percentage of the time.

The output of the FOM (figure 4.9) and the output of DSC (figure 4.13(a)) were seen to be very similar. The main differences exist in the impulse-like features. The second stage smoother appeared to have produced a correction term which re-introduced some of the features eliminated by the FOM. However, they were still highly attenuated in comparison with the input sequence.

A SOM produced an output sequence, y(n), which was, predominantly highly distorted. If this had been the first stage of a dual stage smoother, the difference signal, w(n), would have been highly active. Smoothing the difference sequence produced a correction sequence which, when added to the output of the SOM, y(n), reduced the extent of the distortion. If the second stage was another SOM, the corrective sequence would be zero and therefore have no effect upon y(n). Conversely, if a FOMA was employed, it would have little effect upon the difference sequence, w(n), and therefore the final output signal, z(n), would be almost identical

to the original sequence. Thus the fourth algorithm investigated (DSD) consisted of a SOM as the first stage smoother which would yield a highly active difference signal and a FOM in the second stage to smooth the difference signal.

The periodic sections of the output sequence were severely distorted with many epochs showing extentions/contractions of over 50%. However, the erratic sections were similarly smoothed to that produced by a SOMA.

4.3.2 Informal Subjective Appraisal

The speech synthesised employing the unprocessed coded speech files (CSF) (the data input to the smoothing algorithms) was of high quality and intelligibility. The description of distortions in the speech synthesised utilising processed CSFs are therefore relative to that produced employing the unprocessed CSFs.

The processed CSFs output of the smoothing algorithms were utilised as the input files for the speech synthesis algorithm described in chapter 2. The synthesised speech was output, bandlimited (300 to 3400Hz) and informal subjective comparisons of the synthesised utterances were conducted.

The quality of speech synthesised using the epoch sequences output from the linear smoothing algorithms was noticably different to that for the non-linear smoothers. The speech synthesised from the epoch sequence output of the SOM smoother was totally unintelligible.

During informal listening, one listener who was not familiar with the sentence was able to discern the word "telephone" after a great deal of concentration and several repeats of the sentence.

The speech synthesised from the FOM smoothed epoch sequence was intelligible but sounded gargled, particularly if originating from a female utterances. The female fricative /s/ of 'yes' was masked by noise. The male voiced sounds had a rough, dry throated, husky quality which affected the accent of the speaker but not the intelligibility, while the plosives were indistinct. Nasal sounds within both the male and female utterances appeared to be less distortion than the 'main body' of the complete utterance.

The epoch sequence output from the FOMA smoother, when synthesised, produced speech which was grainy and fuzzy but still clearer than that produced using the FOM epoch sequence. The male utterances contained prominent low frequency components which were probably due to excessive pitch period distortion. The female frication of 'yes' was severely distorted and was perceived as impulse 'clicky' noise. The plosives and nasal consonants were well preserved.

The synthesised speech from the SOMA smoothed epoch sequence was very similar to that of the FOMA. After informal listening, one listener did state that they thought its quality was marginally better than that of the FOMA.

The speech synthesised using the epoch sequence output from the dual stage smoothers DSA and DSB were subjectively identical to that produced by the FOM smoother. The major differences existed in

the segments of erratic epochs which correspond to stop consonants and fricatives and transitions from one to another. In such regions, the synthesised speech from FOM smoothing was noisy and frication were perceived as noise. Unless gross distortions have occurred within the corresponding epoch segments output from DS1 and DS2, the differences in quality are extremally difficult to distinguish.

The speech synthesised using the epoch sequence output from DSC was intelligible and speaker recognition was possible. However, the quality was similar to that produced by a dirty gramophone record: random clicking and noisy background. During some of the male utterances, the speech had a nasal quality to it. The plosives of 'Balam' and 'Walam' were very robust to the distortions. The nasal sounds were severely distorted by clicky, scratchy sounds. During the female utterance of 'yes', the frication /s/ was highly distorted yet the /c/ of 'can' was easily distinguished.

The final section of speech, synthesised from the epoch sequence output from DSD, was highly distorted and not very different to that produced by the second order median smoother.

In retrospect perhaps it is not surprising that a significant reduction in speech quality/intelligibility occurred. Licklider [69] found that the locations of the real zeros of the speech waveform were very important. The smoothing algorithms introduced distortions, in some instances severe, of the epoch durations which causes the distortions of the location of the waveforms real-zeros during synthesis.

4.3.3 Power Spectral Density Measurements

The original speech was unavailable and therefore a segment of speech (figure 4.15(a)), synthesised from the unprocessed CSF (figure 4.1(c)), was utilised to compute a power spectrum (figure 4.15(b)) with which generalised comparisons of power spectra, computed from speech synthesised using processed CSF, were conducted. The aim of this section is, in general, to highlight the changes in the power spectra due to epoch duration sequence smoothing. For this purpose the examples given were selected because the synthesised speech from which the power spectral density plots (PSDP) were calculated differed in quality and intelligibility.

When the epoch sequences were input to the speech synthesis algorithm, the distortions in the epoch duration sequence due to the smoothing algorithm manifested as time distortion of the waveform. These resulted in an increase and/or decrease in the power spectrum over bands of frequencies. However, due to the non-linear nature of some of the smoothing techniques used and the transformation from TES domain to time domain these effects cannot simply be catagorised. The time distortions were a result of both the data and algorithms which cannot be compensated for by post-filtering.

Investigations using differential discrimination of changes in the formant amplitudes and frequencies using synthetic vowel sounds have been reported by Flanagan [70,71]. It was demonstrated that a change of approximately 1.5 dB in the amplitude of the first formant and a 3 to 5% change in formant frequency were just discrimminable.

Due to the resolution of the power spectral density plots (PSDP), changes of less than 2dB in amplitude and 10% in frequency are difficult to resolve. Therefore by comparing the PSDP with that of the original speech it can be stated, with near certainty, that any differences observed exceed the 'just discrimminable' levels. This gives some indication as to the source of some of the distortions perceived.

It was generally observed from the PSDP's that the higher frequencies (greater than 3.5kHz) had been distorted to a greater extent than the lower frequencies (less than 1.5kHz). In fact the first and second spectral peaks, although distorted in amplitude and/or frequency, had not incurred distortions of the same magnitude as that of the higher frequencies. Distortions in the higher frequencies generally varied between 10 and 20dB over complete bands of frequencies and not in isolation, which was the case of distortions in the lower frequencies.

If we consider the distortions introduced by the smoothing algorithms, it is not so difficult to understand the causes of the trends indicated above. Within the TES domain those epochs distorted (increased or decreased in duration) are affected by varying percentages depending on their original length, irrespective of the incremental change. For example, consider epochs of duration 0.5 and 1.25 ms. Both are increased by 0.25 ms thus having durations of 0.75 and 1.5ms. The percentage change being 50% and 20% respectively.

Given a sequence of epochs produced by sampling speech at 20kHz it is not strictly true to say that an epoch of N ms in duration,

extracted from the sequence, would produce a component at 1/2N kHz within the power spectrum because of inter-epoch relationships. However, it does help to give an indication as to why the higher frequencies experience the greater distortions. Applying this relatively simplistic approach, the epoch distortions described previously would result in the original components of 2.0kHz and 0.8kHz being distorted to 1.3kHz and 0.6kHz, changes of 0.7kHz and 0.2kHz, respectively.

As stated earlier these distortions are not predictable and may occur across the spectrum, the cummulative effect being an increase and/or decrease in power spectrum over bands of frequencies.

Figure 4.16(b) is the PSDP for the speech synthesised from the CSF processed by the SOMA smoother, figure 4.16(a). The reason for the prominent low frequency components, described in section 4.3.2, are very clear. The first and second spectral peaks were distorted, and inparticular, the second spectral peak was attenuated (by approximately 10dB) and decreased in frequency. The spectrum below 600Hz was highly attenuated yet the speech was still judged to be highly intelligible.

The speech synthesised from the CSF processed by the SOM smoother, figure 4.17(a), was judged to be totally unintelligible. The PSDP computed from the segment of speech of figure 4.17(a) is given in figure 4.17(b). Compared with figure 4.16(b) it is clearly observed that the first and second spectral peaks had merged, the resulting spectral peak had also been increased in frequency. The majority of the spectrum below 800Hz was attenuated while at frquencies

greater than 1.0kHz the spectrum had been amplified. Such significant spectral changes resulted in the speech being unintelligible.

Finally, figure 4.18(b) is the PSDP for the speech of figure 4.18(a), which was synthesised from the CSF processed by the third dual staged smoother, DSC. Comparing figures 4.18(b) and 4.17(b) it is observed that the spectra are very similar. However, the speech output from DSC was intelligible and speaker recognition was possible whereas the speech output from the SOM smoother was totally unintelligible. It was therefore presumed that the epoch duration sequences these power spectra related to had experienced very similar distortions when smoothed and this resulted in the similar spectra on synthesis.

4.4 Conclusions

The smoothing algorithms investigated were found to be effective in the elimination and/or attenuation of the erratic and 'impulse-like' features within the epoch sequences. In so doing, the smoothers introduced significant distortions into the periodic epoch duration sequences.

It was observed that the Second Order Median smoother yielded the greatest smoothing but the speech synthesised from the resulting epoch duration sequence was unintelligible.

The First and Second Order Moving Average smoothers output the least attenuated sequence and the speech synthesised from this was intelligible though noticeably degraded.

The First Order Median and Dual Stage smoothers output differed in the extent to which smoothing was achieved and hence the quality of synthesised speech.

These investigations have sought to demonstrated that only modest levels of smoothing may be applied to the epoch duration sequences before significant degradation in speech quality occurs.

Since the smoothing algorithms were investigated as possible pre-processors to enhance the effectiveness of other data reduction techniques, it must be concluded that a pre-processor of the epoch duration sequence, involving simple numerical smoothing, is of little value for it appears to degrade the quality/intelligibility of the synthesised waveform in an unacceptable manner. However, it has been suggested that median smoothing may be a useful tool if a more sophisticated epoch classification technique were to be adopted [72].

SEGMENT PROCESSED	ALGORITHM	ALGORITHM OUTPUT
1(a) 1(c)	—	—
1(a) 1(c)	1st. Order Mov. Ave.	7(a) 7(b)
1(a) 1(c)	2nd. Order Mov. Ave.	8(a) 8(b)
1(a) 1(c)	1st. Order Med. Filt.	9(a) 9(b)
1(a) 1(c)	2nd. Order Med. Filt.	10(a) 10(b)
1(a) 1(c)	Dual Stage Smoother A	11(a) 11(b)
1(a) 1(c)	Dual Stage Smoother B	12(a) 12(b)
1(a) 1(c)	Dual Stage Smoother C	13(a) 13(b)
1(a) 1(c)	Dual Stage Smoother D	14(a) 14(b)

Table 4.2 : Cross reference of figure numbers and algorithms.

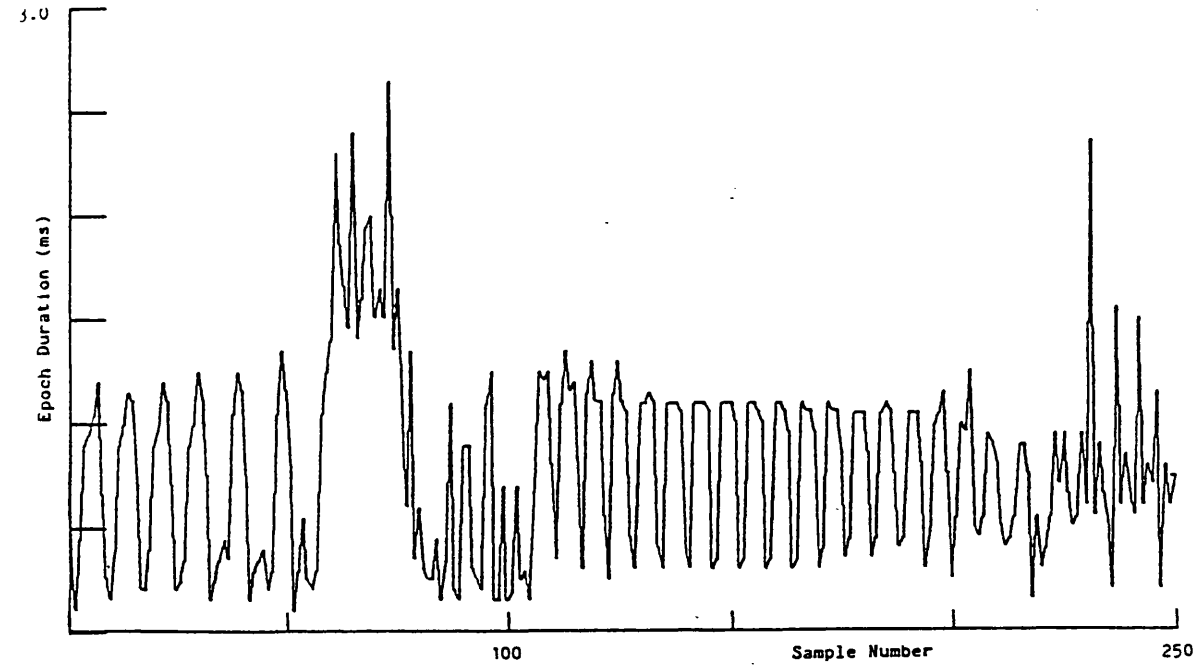
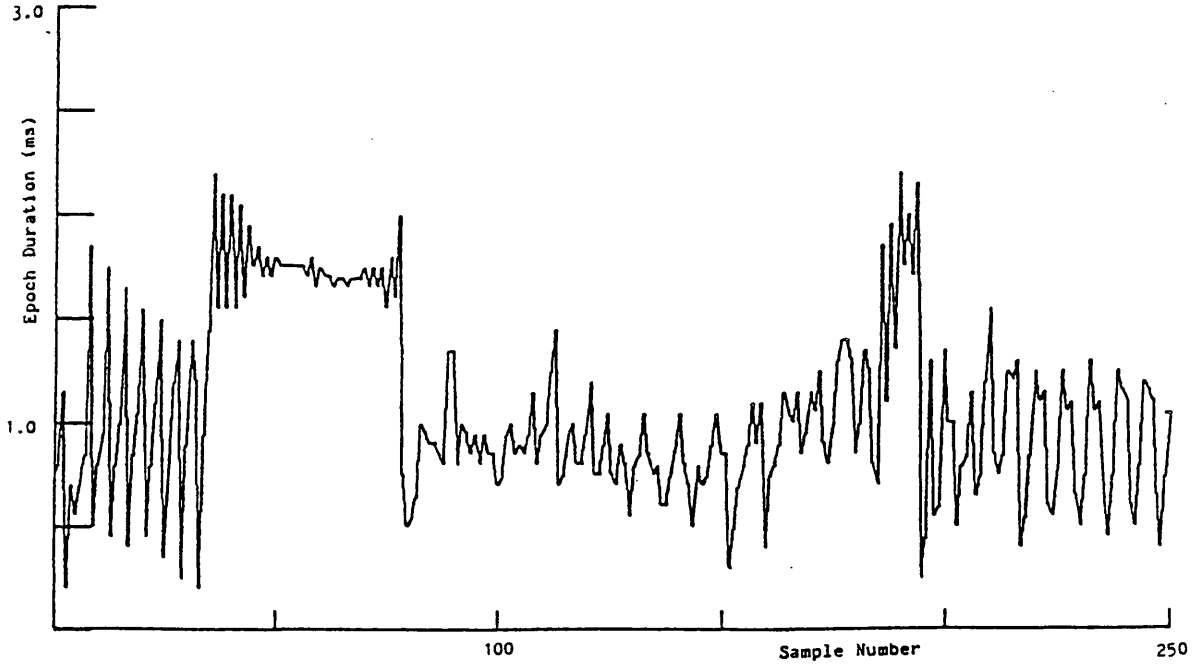


Figure 4.1(a),(b) : Sequential plots of Epoch duration output from Al-Doubooni's TES coder.



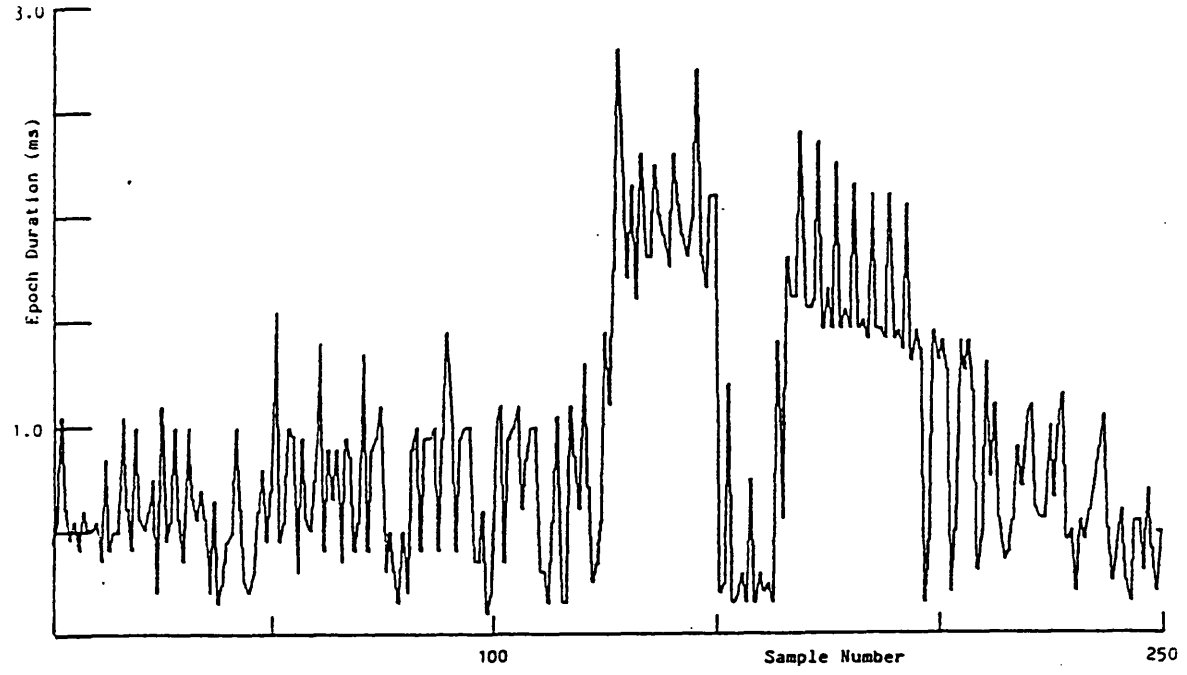
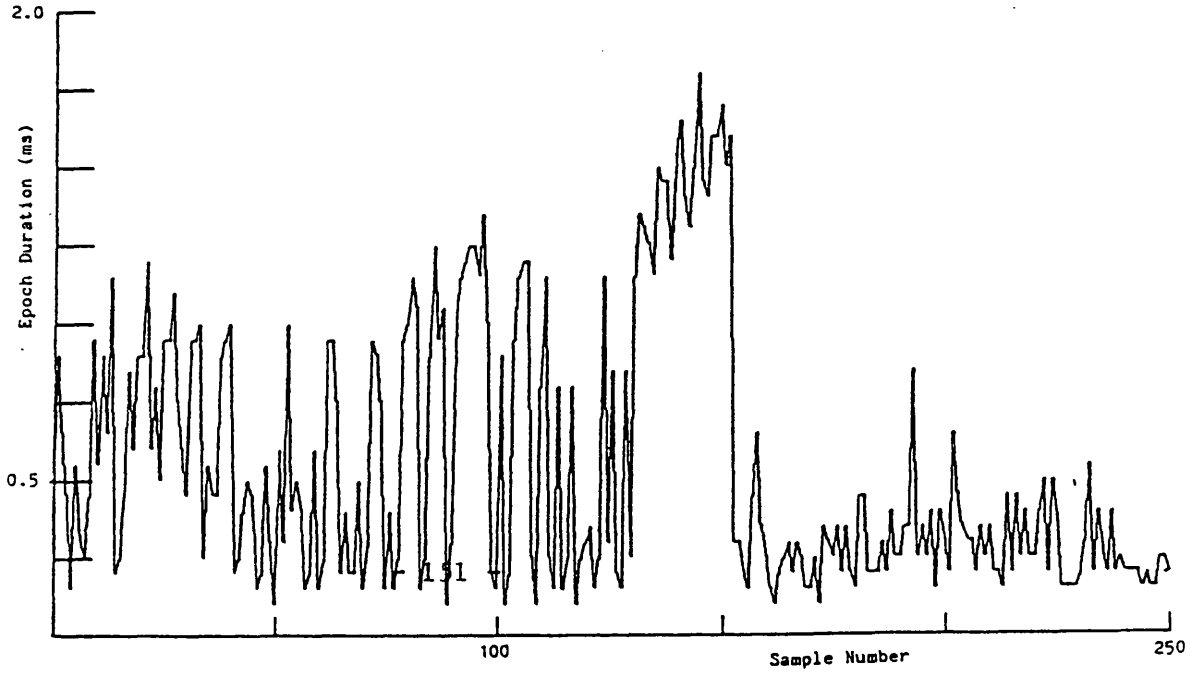


Figure 4.1(c),(d) : Sequential plots of Epoch duration output from Al-Doubooni's TES coder.



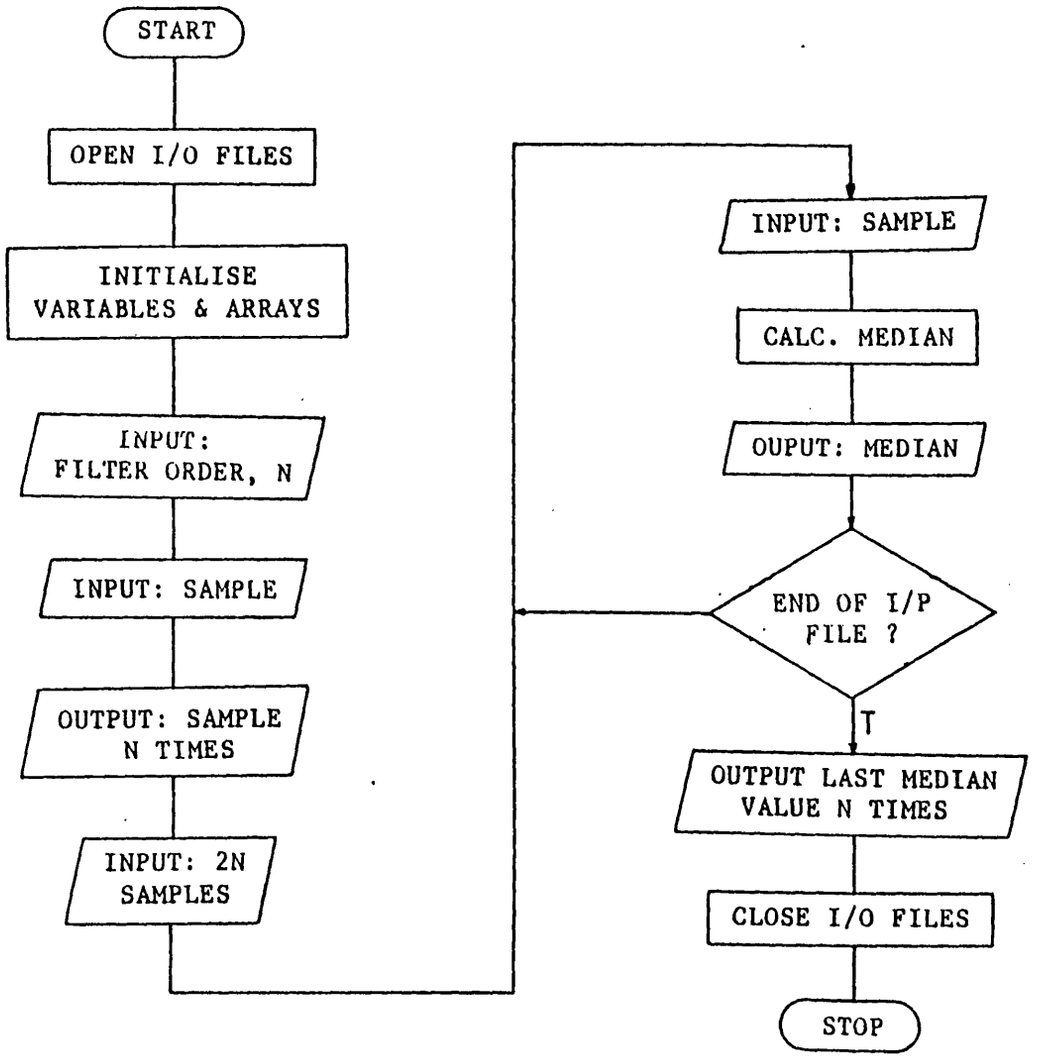


Figure 4.2(a) : Flow chart of the Median Filtering Algorithm.

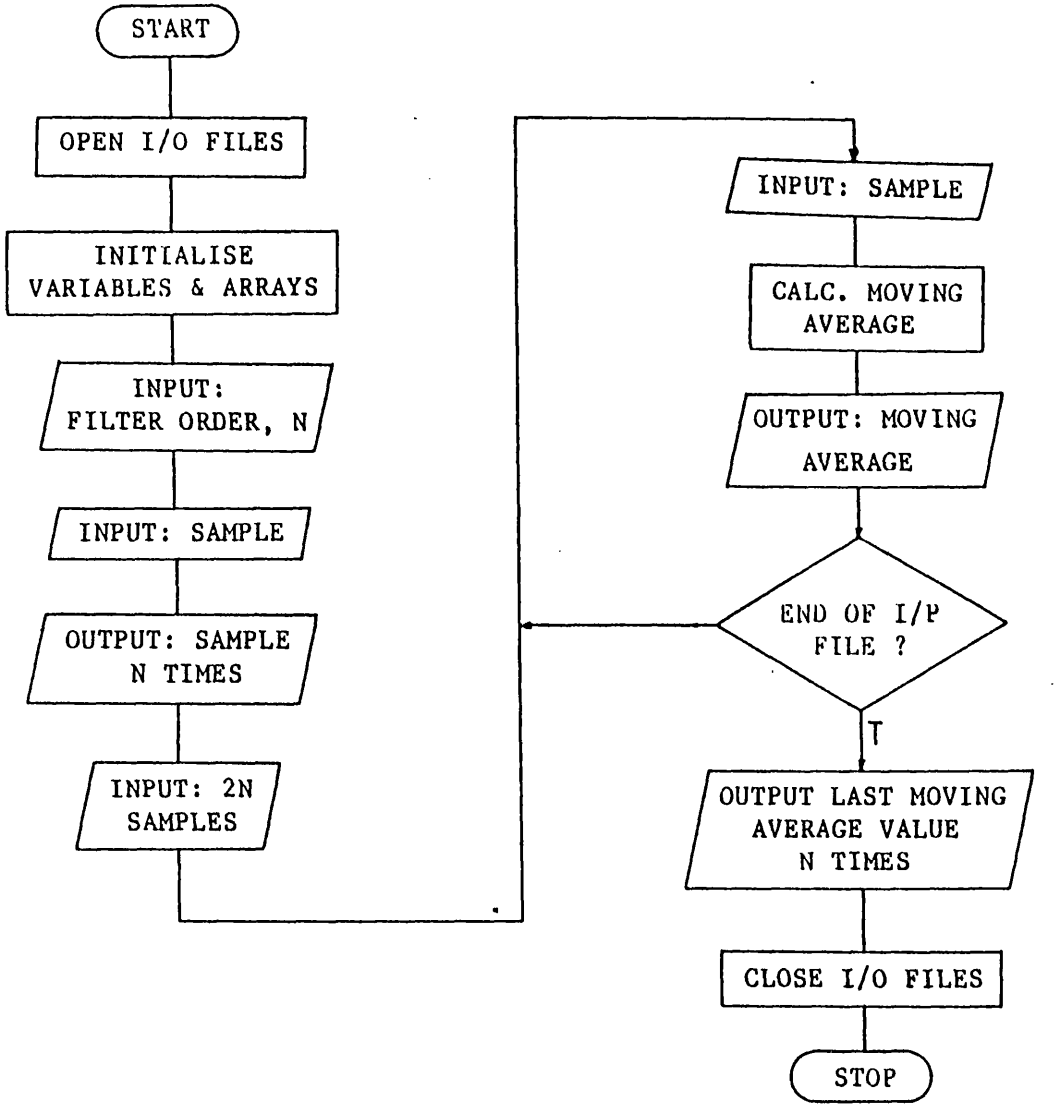


Figure 4.2(b) : Flow chart of the Moving Average Filtering Algorithm.

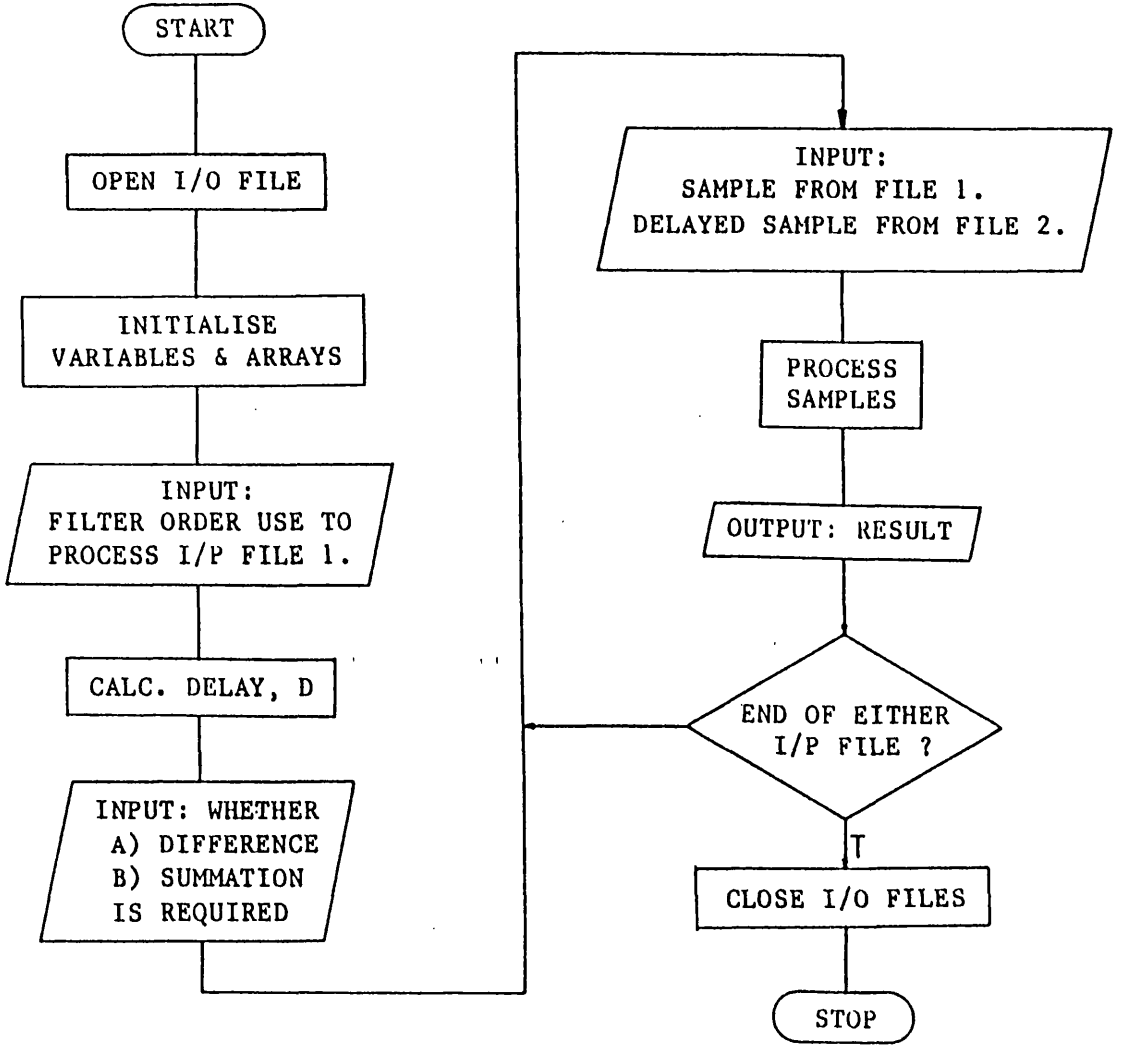


Figure 4.3 : Flow chart of the Dual Stage Smoothing Algorithm.

Figure 4.4 : Black box representation of a Single stage smoother.

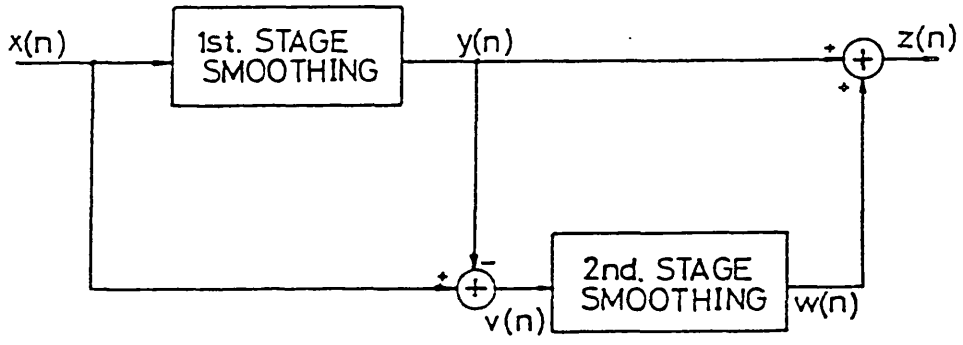


Figure 4.5 : Black box representation of a Dual stage smoother.

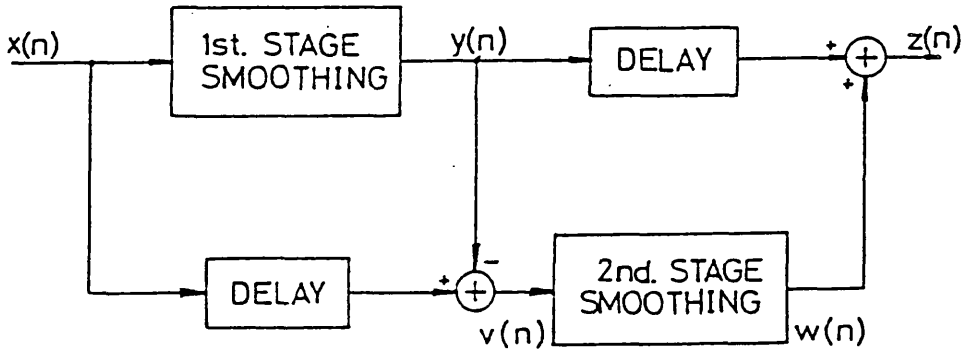


Figure 4.6 : Black box representation of a realisable Dual Stage Smoother.

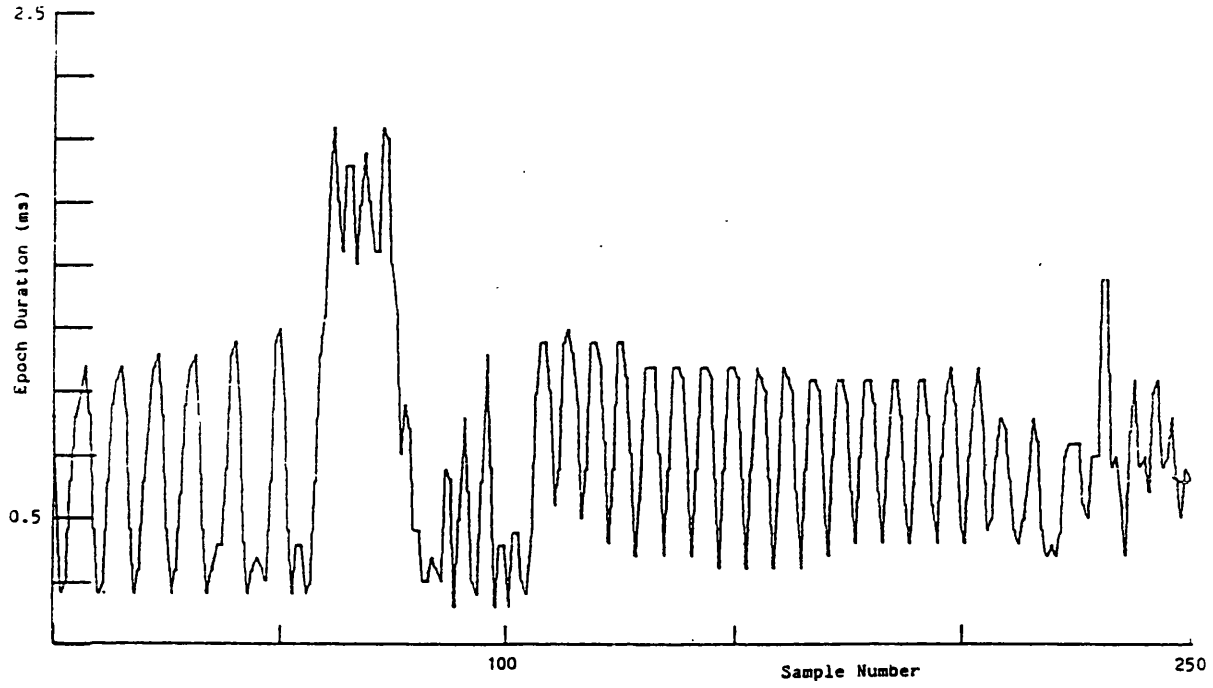
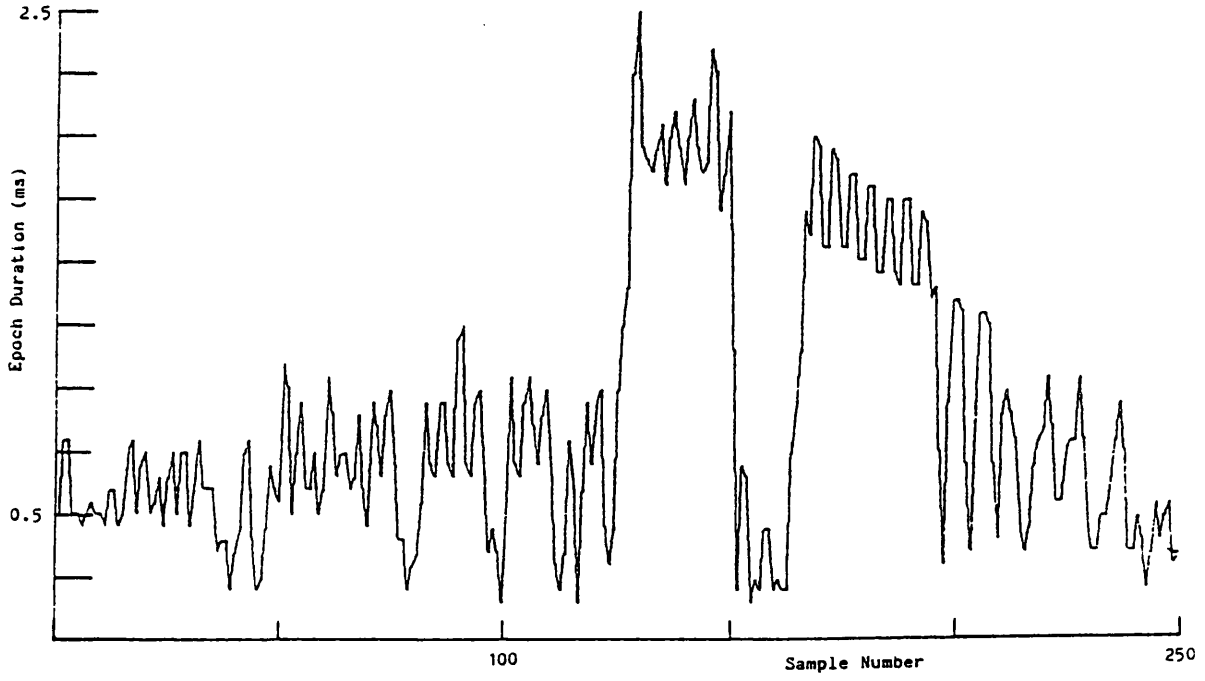


Figure 4.7 : (a) Epoch sequence of Figure 4.1(a) processed by a 1st order Moving Average Filter.

(b) Epoch sequence of Figure 4.1(c) processed by a 1st order Moving Average Filter.



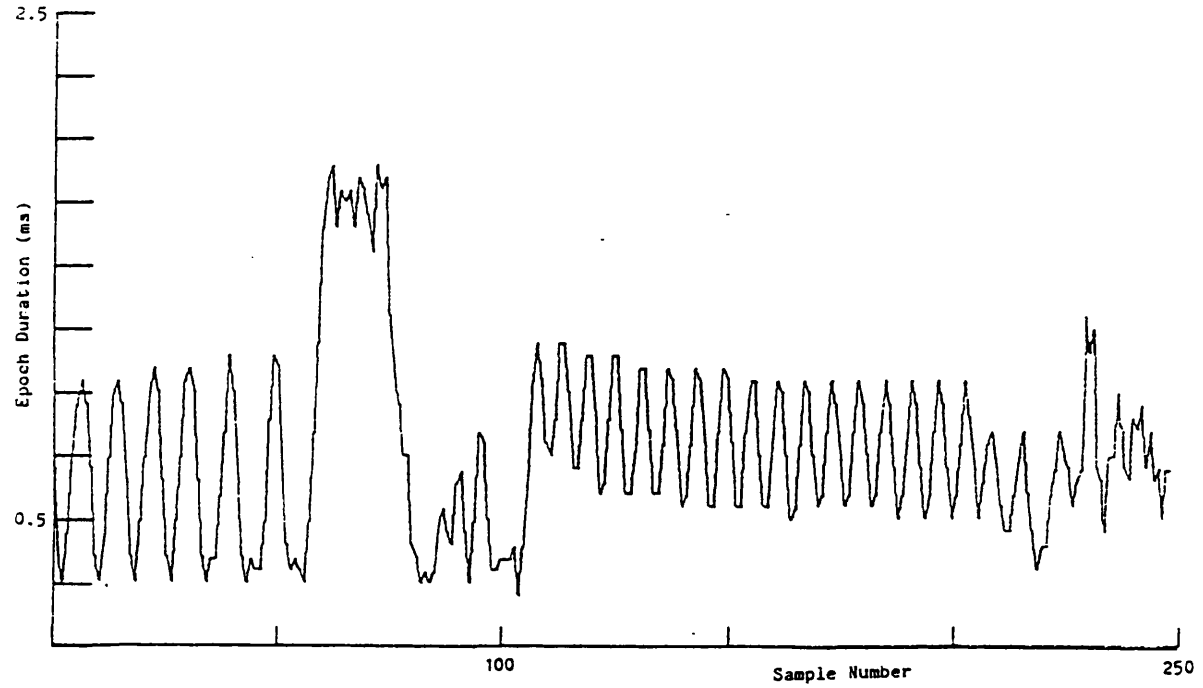
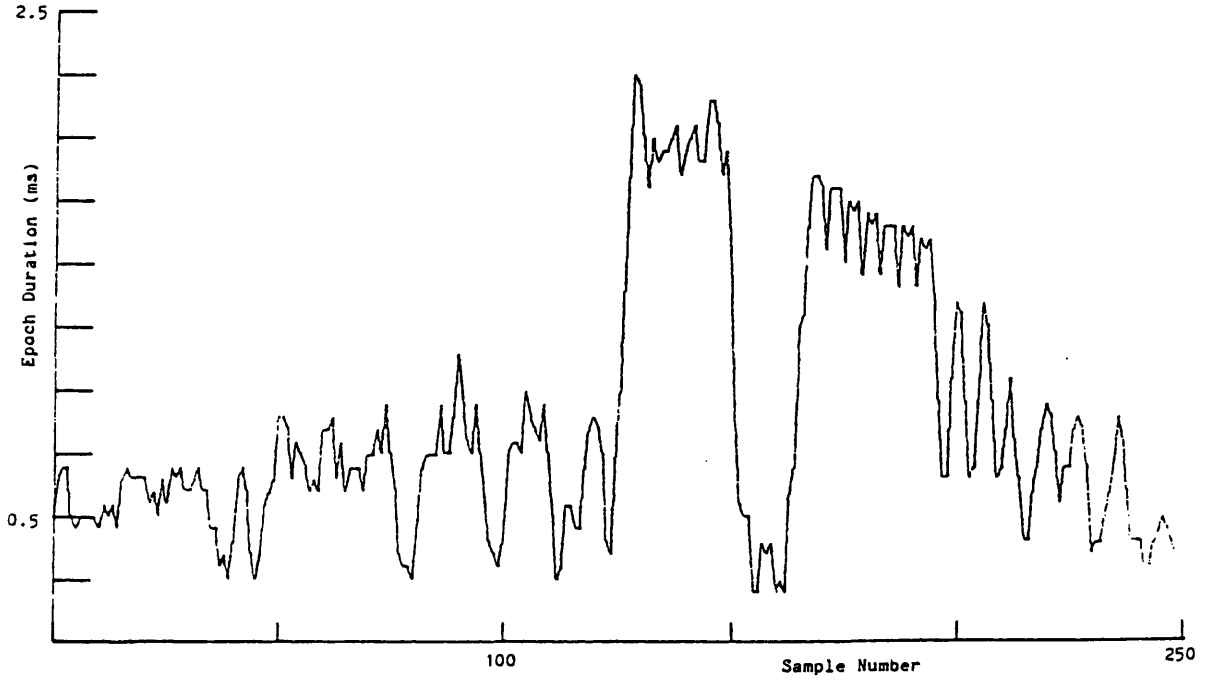


Figure 4.8 : (a) Epoch sequence of Figure 4.1(a) processed by a 2nd order Moving Average Filter.

(b) Epoch sequence of Figure 4.1(c) processed by a 2nd order Moving Average Filter.



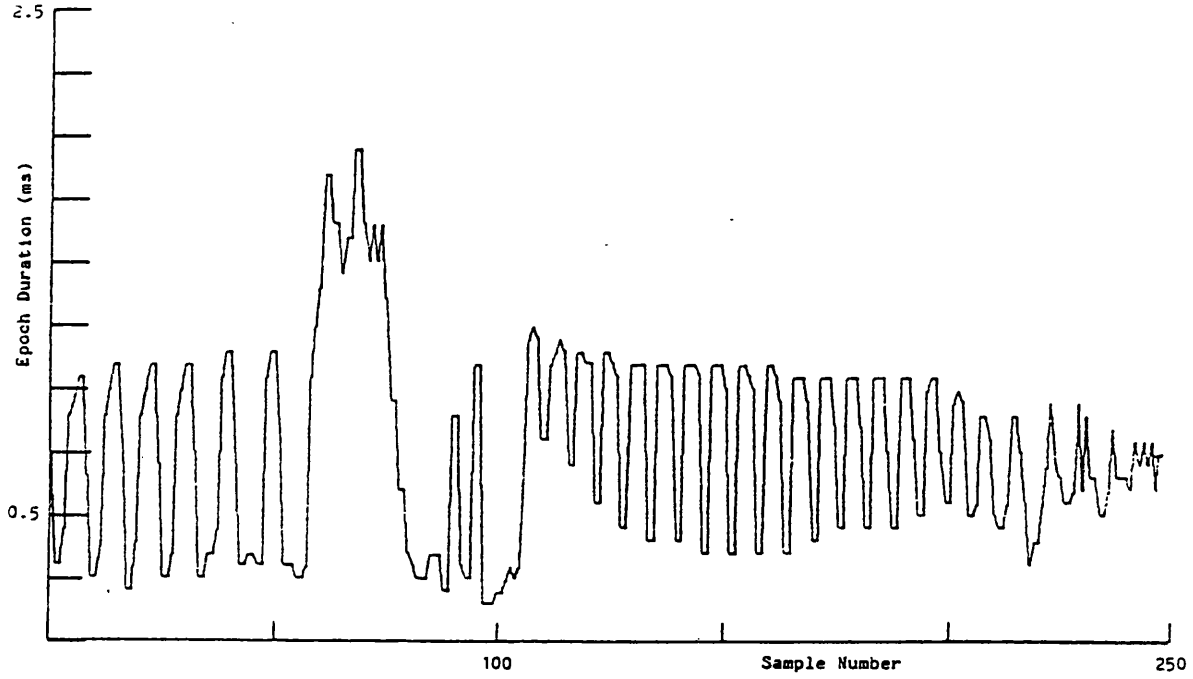
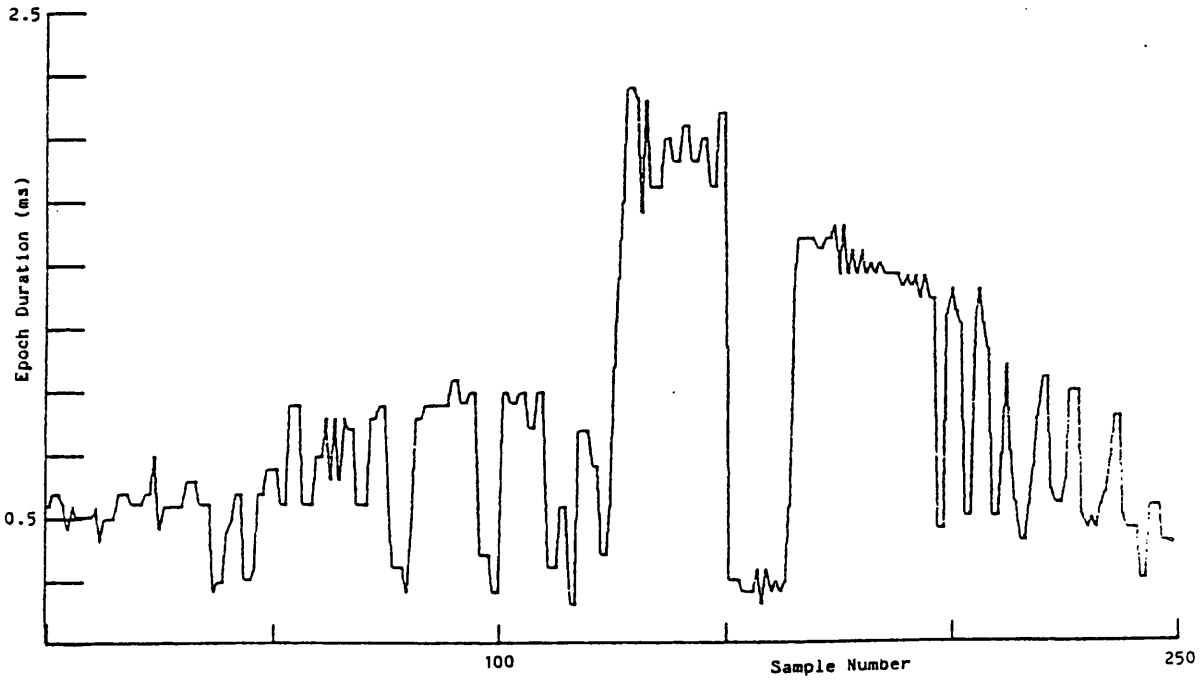


Figure 4.9 : (a) Epoch sequence of Figure 4.1(a) processed by a 1st order Median Filter.

(b) Epoch sequence of Figure 4.1(c) processed by a 1st order Median Filter.



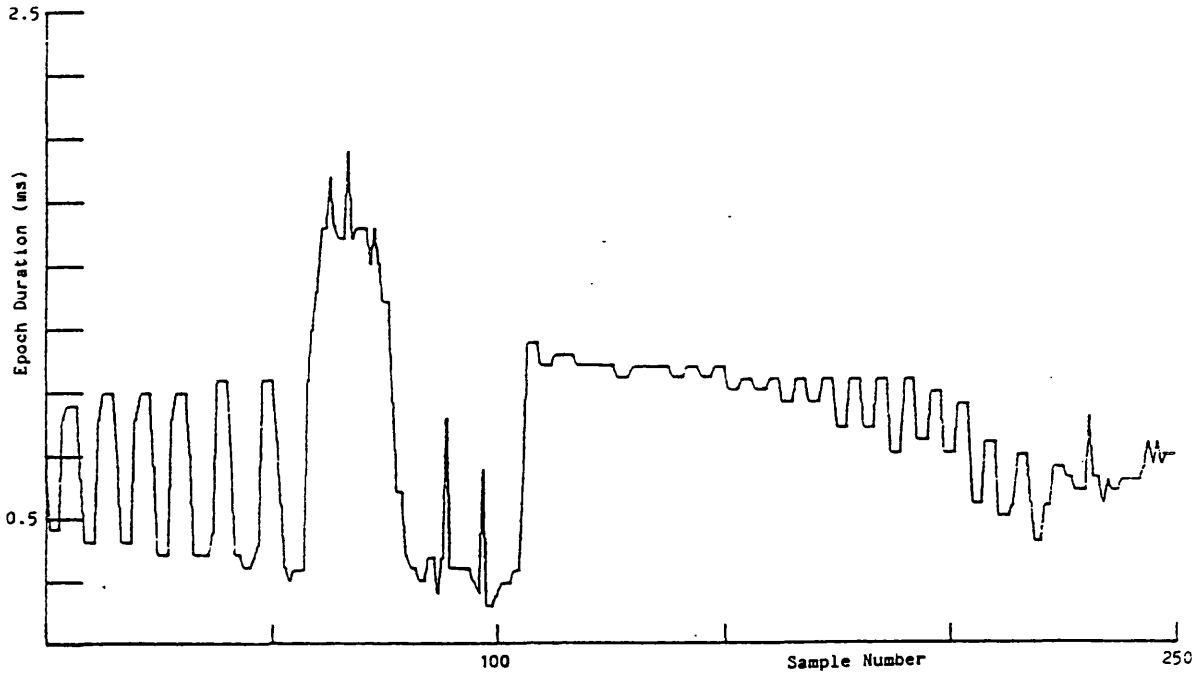
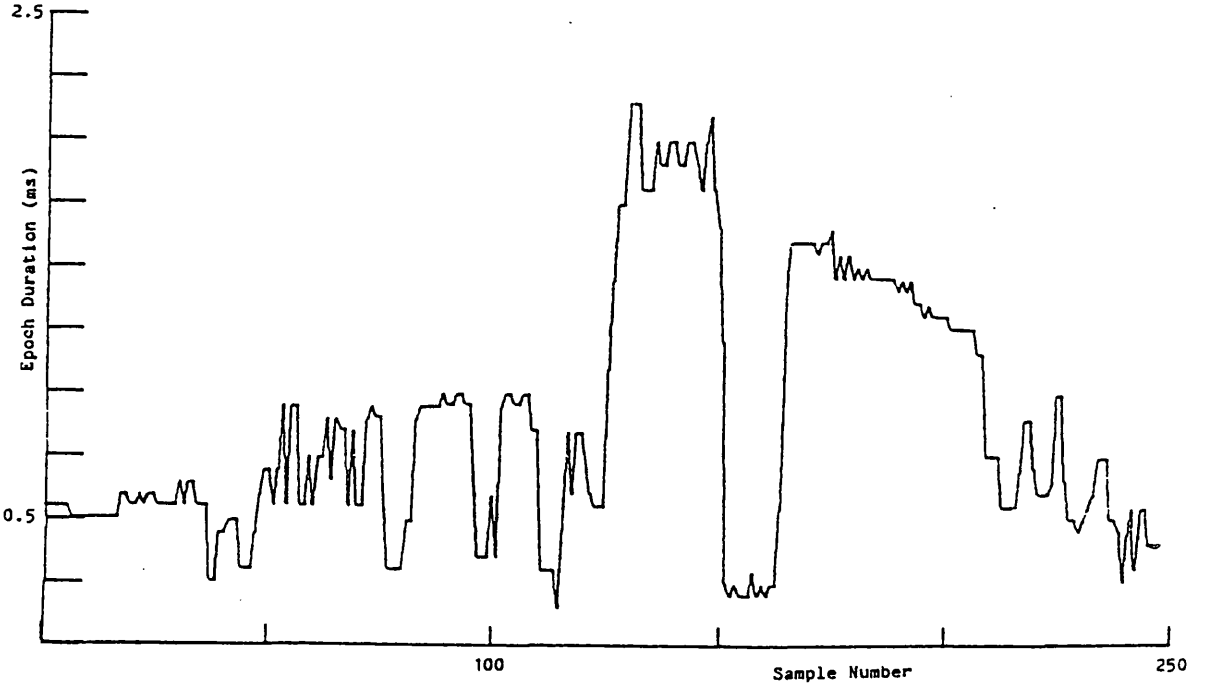


Figure 4.10 : (a) Epoch sequence of Figure 4.1(a) processed by a 2nd order Median Filter.

(b) Epoch sequence of Figure 4.1(c) processed by a 2nd order Median Filter.



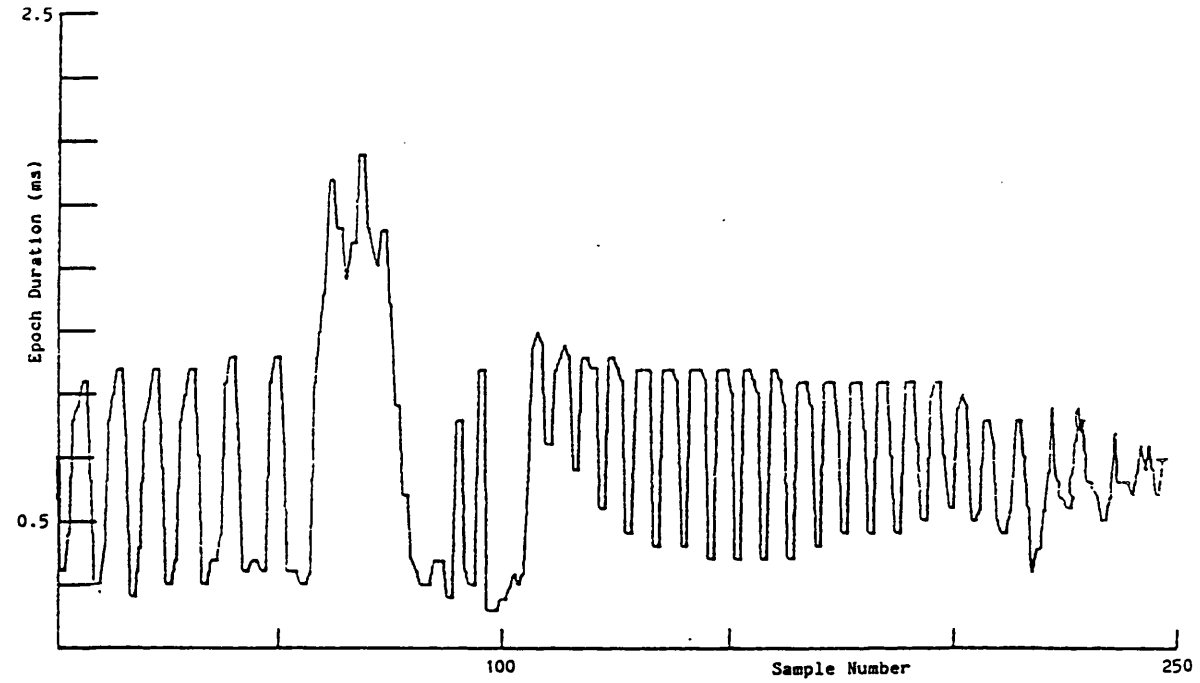
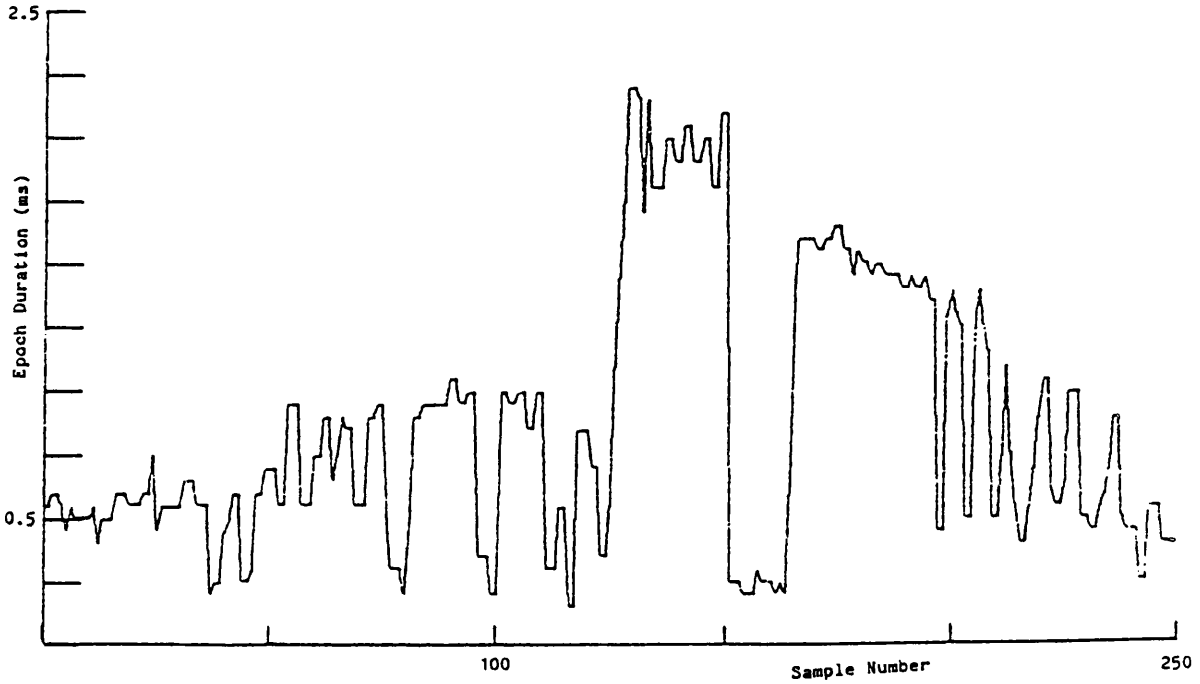


Figure 4.11 : (a) Epoch sequence of Figure 4.1(a) processed by the Dual Stage Smoother A.

(b) Epoch sequence of Figure 4.1(c) processed by the Dual Stage Smoother A.



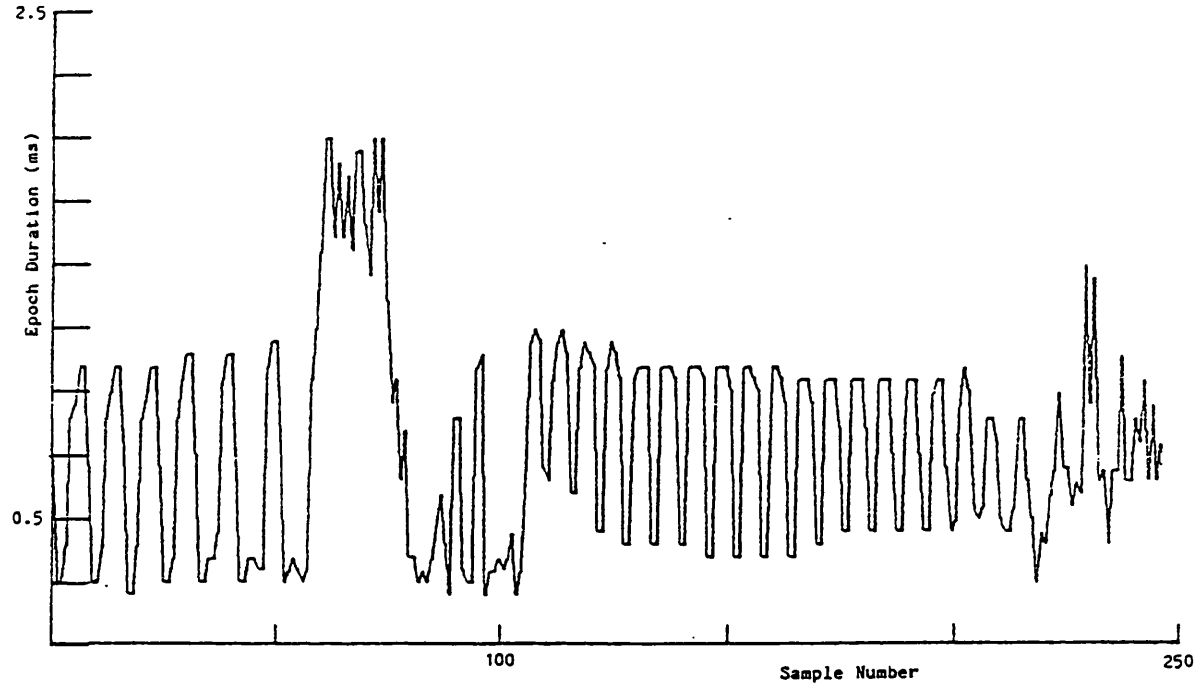
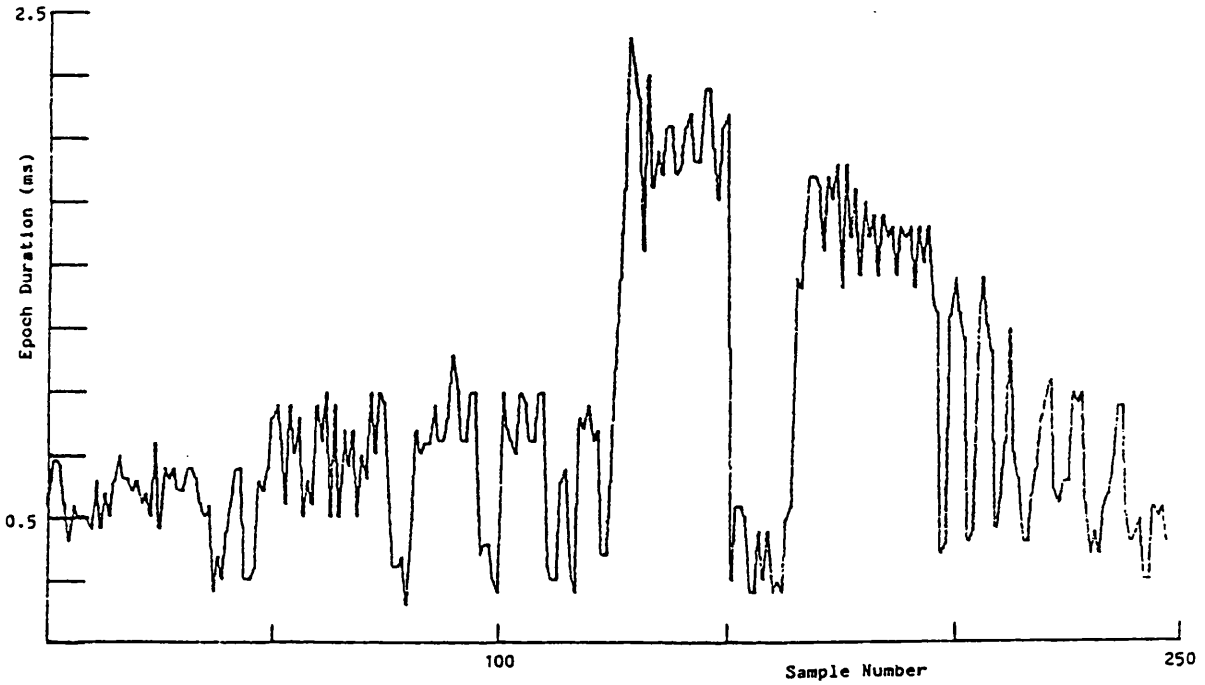


Figure 4.12 : (a) Epoch sequence of Figure 4.1(a) processed by the Dual Stage Smoother B.

(b) Epoch sequence of Figure 4.1(c) processed by the Dual Stage Smoother B.



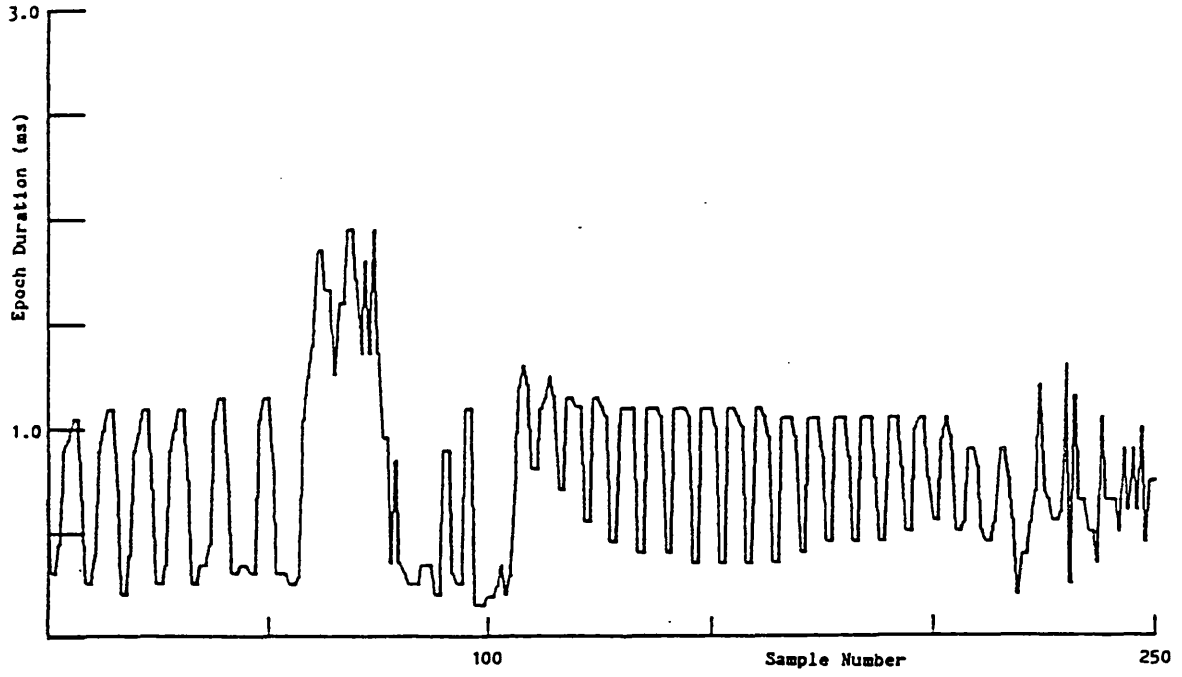
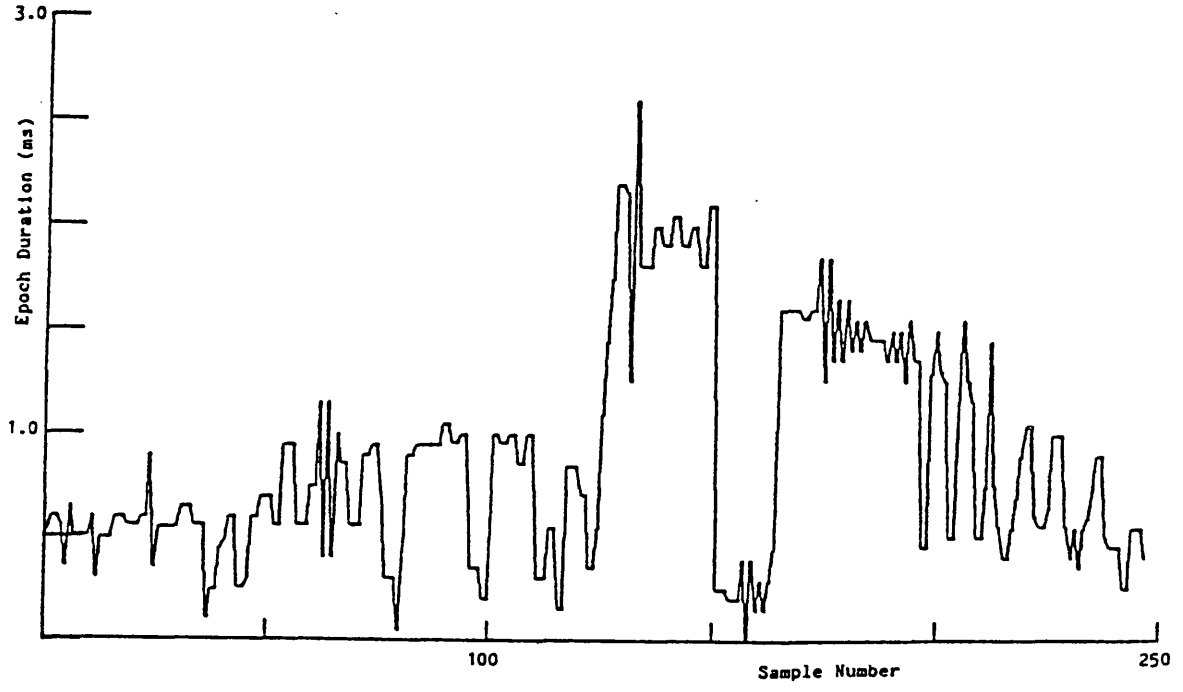


Figure 4.13 : (a) Epoch sequence of Figure 4.1(a) processed by the Dual Stage Smoother C.

(b) Epoch sequence of Figure 4.1(c) processed by the Dual Stage Smoother C.



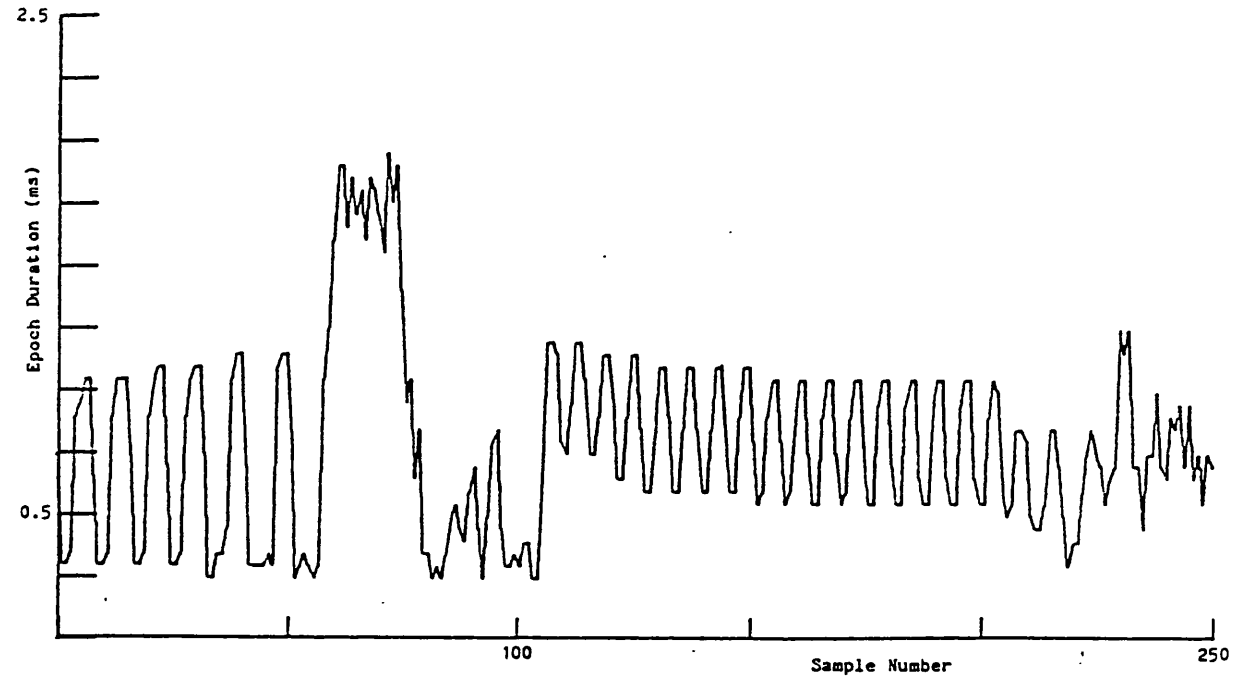
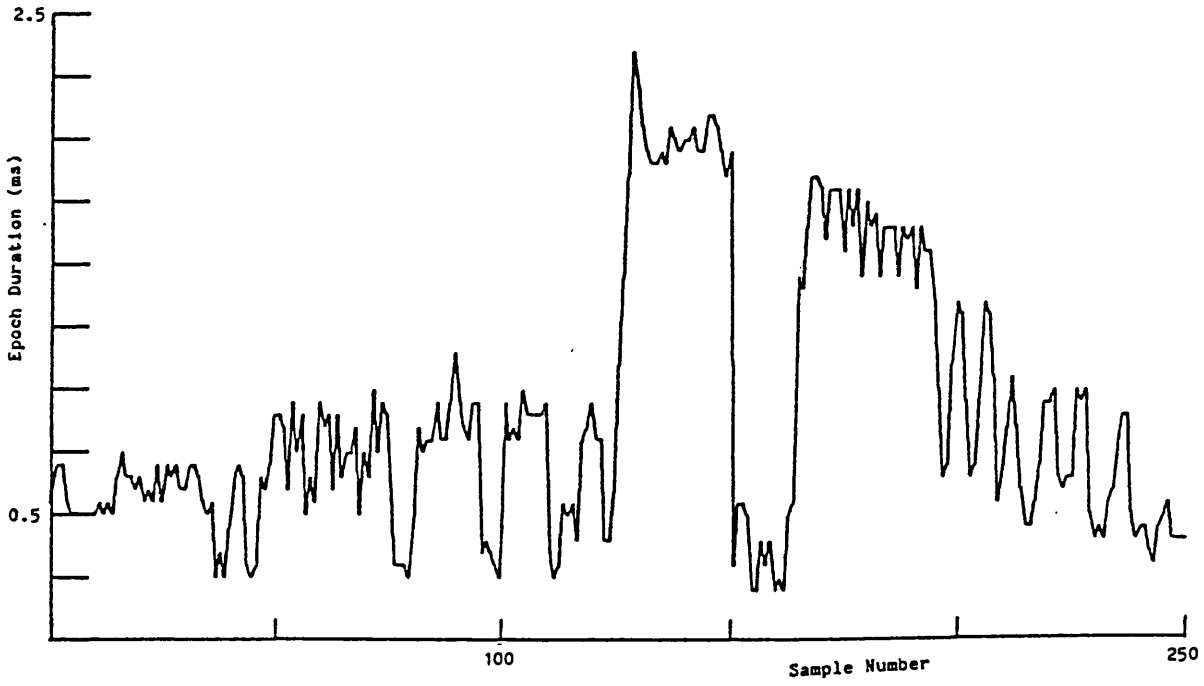


Figure 4.14 : (a) Epoch sequence of Figure 4.1(a) processed by the Dual Stage Smoother D.

(b) Epoch sequence of Figure 4.1(c) processed by the Dual Stage Smoother D.



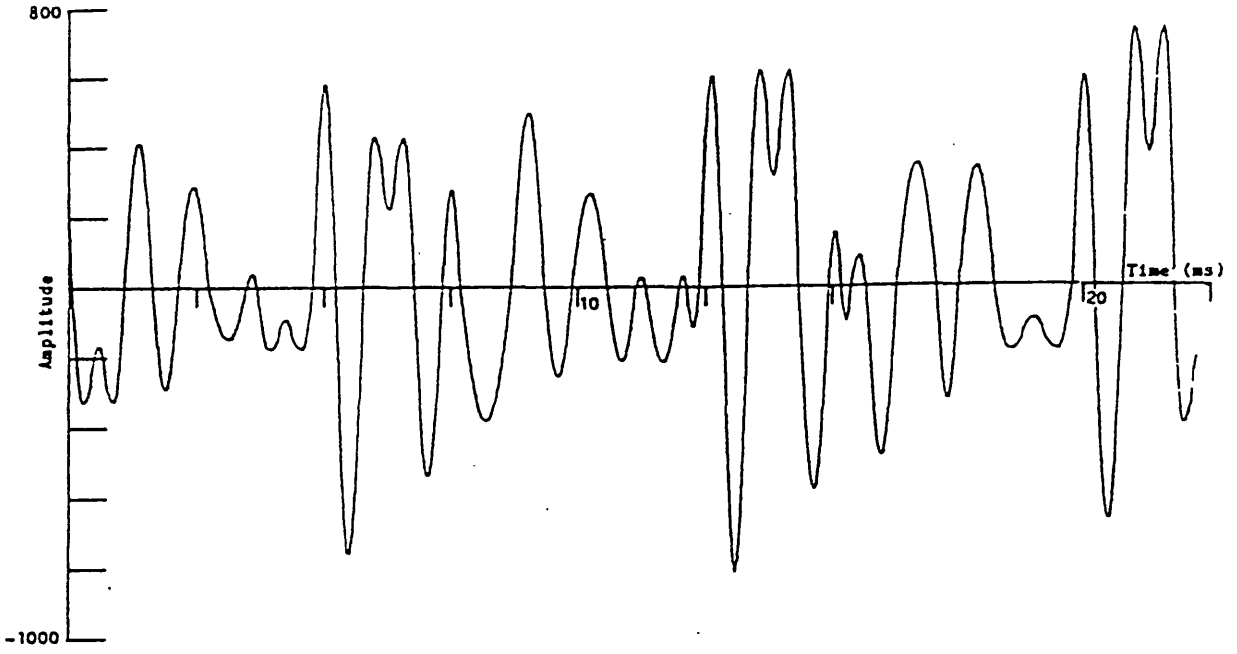
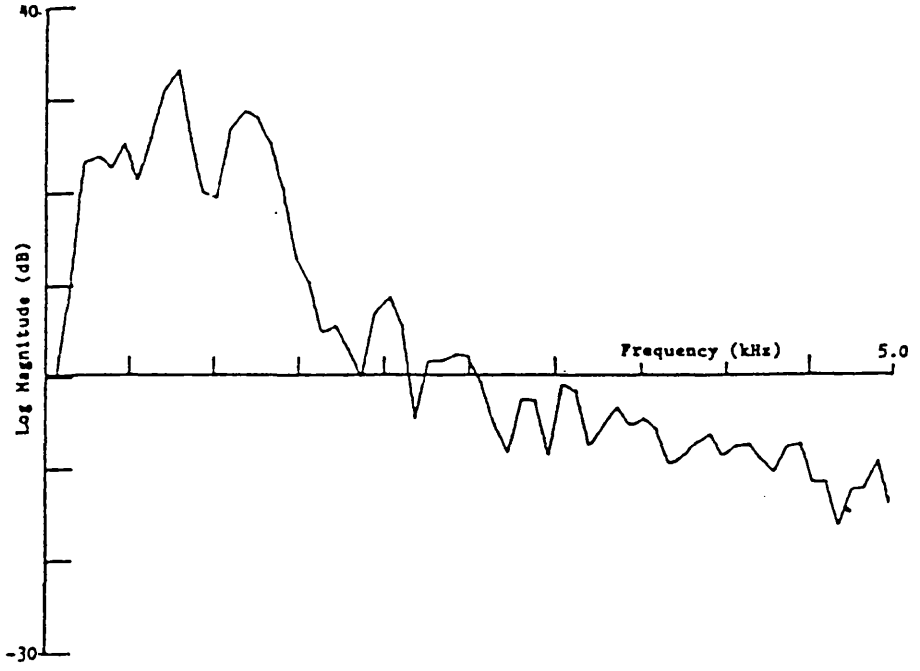


Figure 4.15 : (a) Segment of speech waveform synthesised using the input epoch duration sequence of Figure 4.1(c).

(b) Corresponding power spectral density plot .



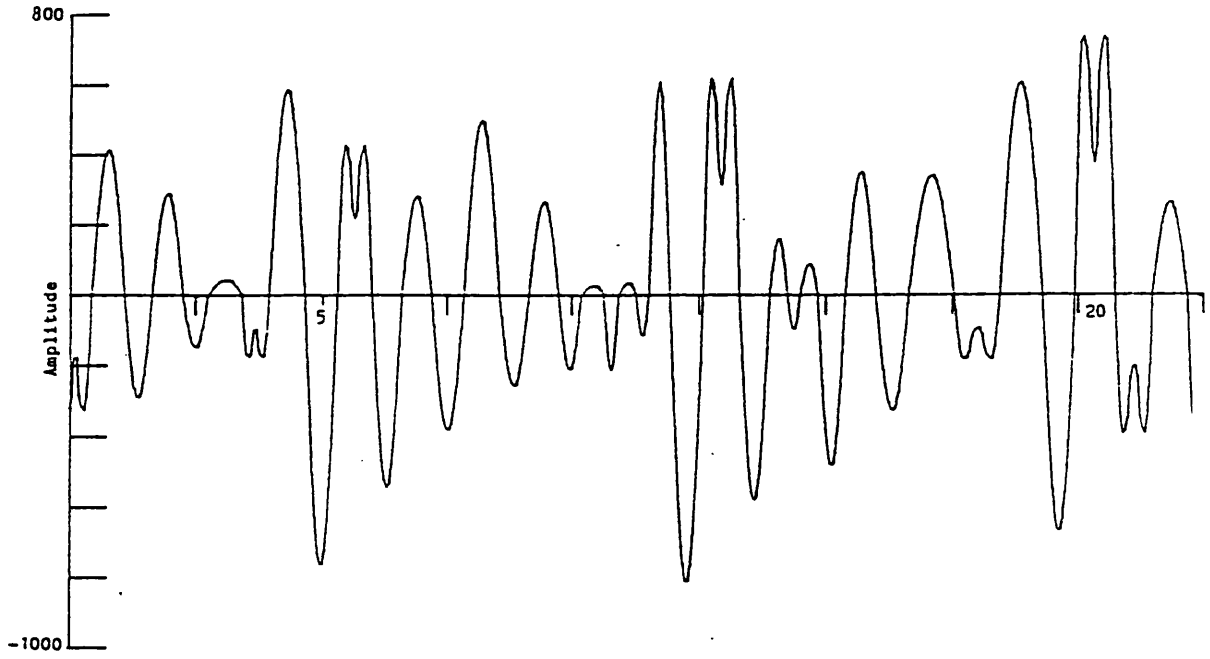
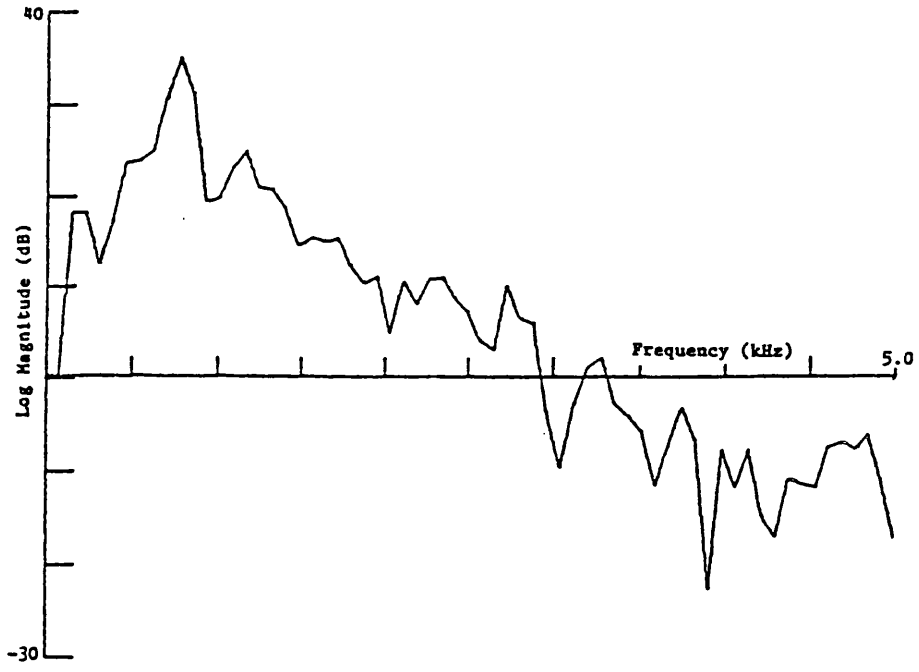


Figure 4.16 : (a) Segment of speech waveform synthesised using the epoch duration sequence of Figure 4.10(b).

(b) Corresponding power spectral density plot.



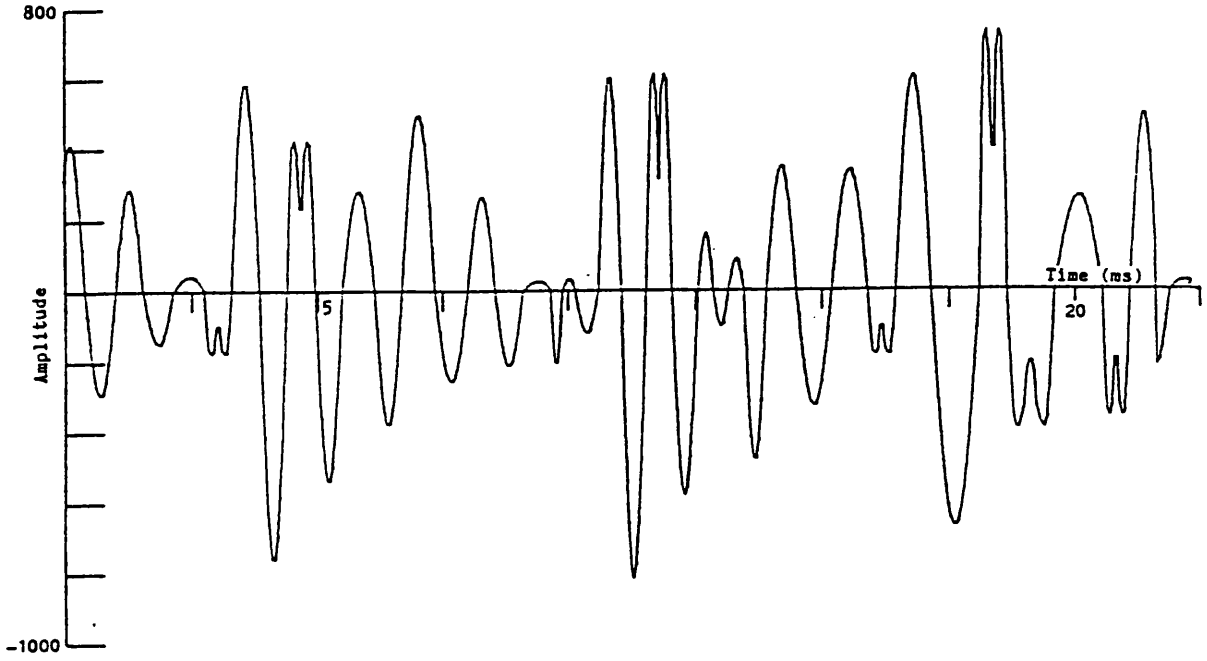
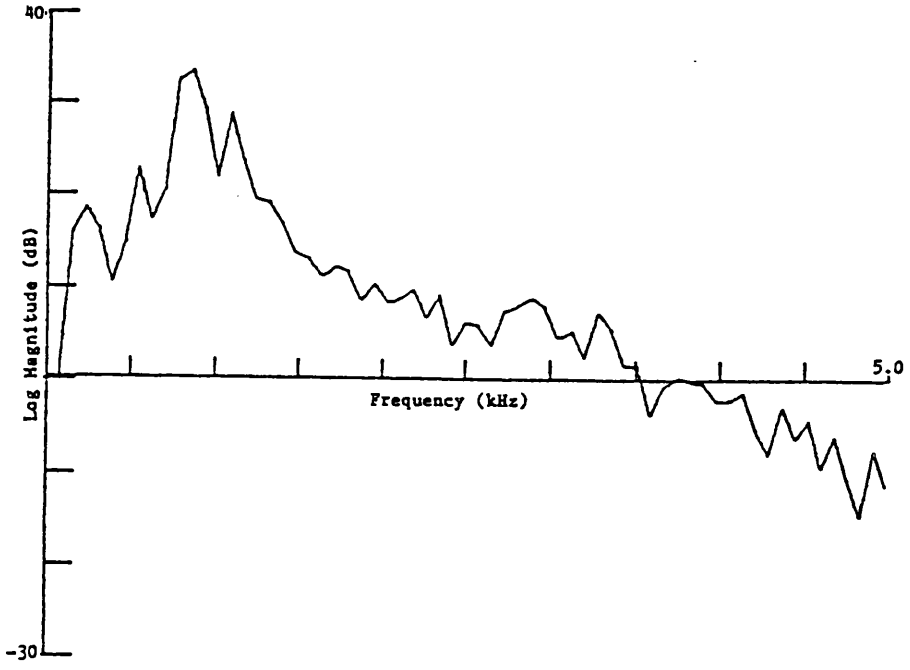


Figure 4.17 : (a) Segment of speech waveform synthesised using the epoch duration sequence of Figure 4.12(b).

(b) Corresponding power spectral density plot.



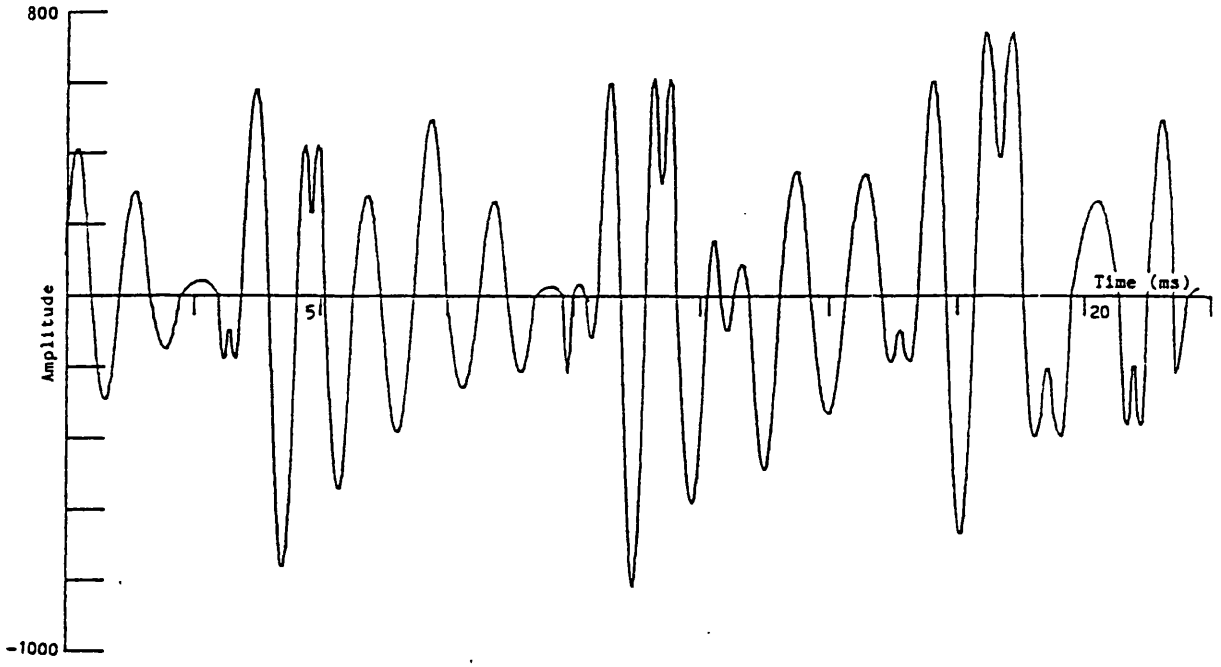
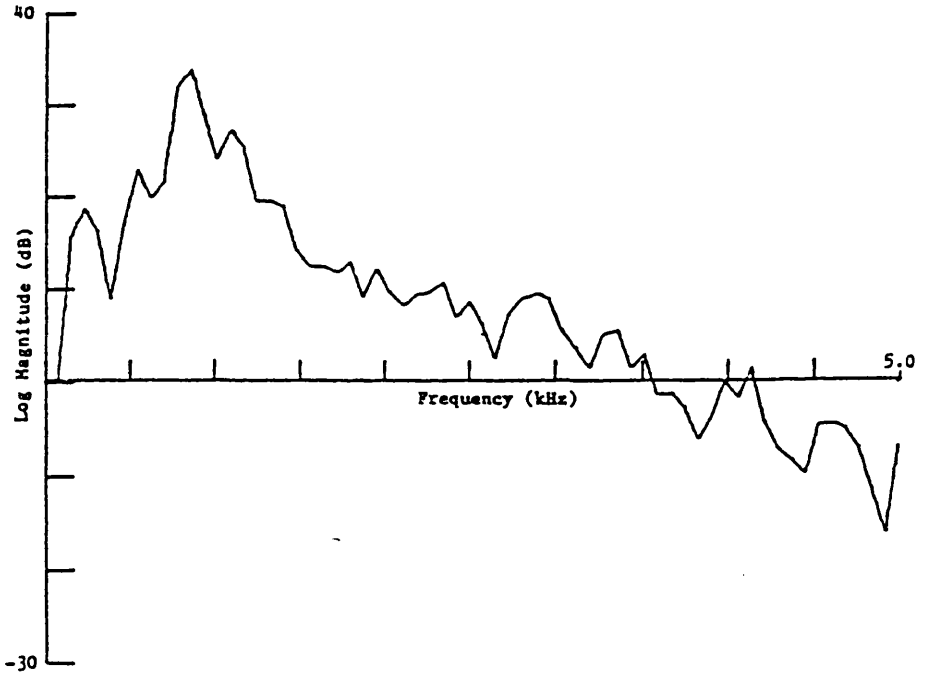


Figure 4.18 : (a) Segment of speech waveform synthesised using the epoch duration sequence of Figure 4.15(b).

(b) Corresponding power spectra density plot.



Chapter 5

Predictors, Interpolators and Extremal Coding

5.1 Introduction

Numerous redundancy reduction and data compression techniques have been investigated by independent researchers for applications within communication systems [73-78]. Their common goal being to ensure that only significant data is transmitted. The techniques reported were, in general, waveform coders rather than source coders. The most effective and widely used of these techniques being polynomial predictors and interpolators.

The polynomial prediction algorithms were based on a finite difference technique by means of which an n-th order polynomial was to be passed through n+1 data points. The polynomial was extrapolated one data point at a time to yield the predicted data. If the next data point fell within a tolerance band (or aperture) about the predicted value, it was rejected as redundant since that data point could be reconstructed (within a specified tolerance) using previous values. If the new data point was outside the aperture then it was transmitted.

A First Order Predictor (FOP) utilises an extrapolation polynomial of the form

$$y(n+1) = 2.y(n) - y(n-1) \qquad (5.1)$$

This extrapolation is a straight line between the last two data points. Initially the first two data points are transmitted and a straight line is extrapolated through them for one data point. An aperture is placed about the predicted point. If the next data

point is within the prediction aperture then that point is not transmitted and the line is extrapolated a further data point. If the next point is outside the prediction aperture then the data point is transmitted and the line extrapolated through the present data point (which is transmitted) and the previous data point. The functional operation of a FOP algorithm is shown in figure 5.1.

Inspection of figures 5.2(a), 5.3(a) and 5.4(a) reveal that, between extrema of an epoch sequence, the epochs appear to lie approximately on a straight line. Pilot investigations were conducted in order to estimate the data reduction achievable using FOP alogrithms on a sequence of epoch durations.

When applied in the time domain, FOP's employed an aperture whose width was a constant value. For the TES domain, a choice of aperture width was difficult because a small aperture ($\pm$0.1ms) meant that the shorter epoch durations had a realistic tolerance band while the larger epoch durations had a much tighter tolerance and this reduced the extent of the achievable data reduction. The converse situation arose when a larger aperture ($\pm$0.4ms) was employed. Here the larger epoch durations had a realistic aperture while that of the shorter epoch durations was excessive. For such an aperture, a data reduction of 1.2:1 was achieved yet the synthesised speech was totally unintelligible.

A possible strategy was to make the aperture a function of the predicted sample or past samples. The simplest function being a percentage of the predicted value. For example, if the predicted value was y(n) then the corresponding aperture would be 0.1y(n) for

a 20% tolerance band. However, unless the tolerance band was greater than 20%, no significant data compression would be achieved since the shorter epochs (.LT. 0.5ms) would have no aperture. With tolerance bands greater than 20%, data reductions were achieved although but the speech synthesised was not intelligible.

It was soon realised that the application of a FOP to the epoch duration sequences would not yield a suitable degree of data reduction while maintaining a high degree of intelligibility in the synthesised speech. Higher order polynomial predictors may yield marginally better results than the FOP but due to the quasi-stationary nature of the epoch duration statistics the synthesised speech would still be of very poor quality.

It was stated previously that interpolators have also been widely used for data compression. Interpolators differ from the corresponding prediction algorithms by the fact that all sample values between the last transmitted value and the present value affect the interpolation. Interpolators are more efficient in environments where the samples are perturbed by low level, high frequency noise [73]. A number of examples of interpolators for data compression are given in the literature [73-76]. Even though the epoch sequences may be thought of as a sequence plus noise (chapter 4), the 'noise' cannot be considered to be low level and, in order to achieve useful data reductions, large apertures would be required. The effect of such an aperture would be similar to that described earlier in case of the prediction algorithms. Therefore, for epoch sequences these interpolation algorithms would not yield better data

compression than that of the predictors.

It was therefore concluded that predictors and interpolators of the type described could not achieve useable levels of data compression. To minimise distortions would result in an increase in the required data due to the rapid changes in epoch sequences.

In 1959 the results of investigations into Extremal Coding for Speech Transmission were reported by Mathews [53]. The paper described how time positions of a waveforms extreme were located and the waveform was reconstructed employing an interpolation function between extrema. The synthesis was such that the original extreme were retained and no discontinuities existed in the waveform nor its derivative at its extrema. The information transmitted consisted of the signal amplitude at the extreme and the time interval between successive extrema. Subjectively, extremal coding was found to require approximately half the channel capacity of companded PCM for equivalent speech quality transmission.

From the work by Mathews it was hypothesised that extremal coding of epoch duration sequences, employing a suitable interpolating function for reconstruction, may yield a significant data reduction whilst retaining a high degree of intelligibility together with reasonable quality of synthesised speech. As previously noted the interpolators and predictors investigated were unable to 'track' the sudden changes of the epoch sequences. However, Extremal Coding does not attempt to predict the extrema but utilises them for the prediction of the 'samples' between extrema. This technique would eliminate the need to transmit the epoch durations which exist between

the extrema or alternatively lie outside some preset tolerance band about a predicted value.

In order to investigate the hypothesis, two sets of algorithms were developed. Such a set consisted of a transmitter and receiver algorithm. The first set employed extremal coding of the epoch duration sequence only and the remaining TES parameters were 'transmitted' unprocessed. This algorithm was developed to isolate the type and extent of the distortions introduced by extremal coding of the epoch duration sequence. The second set of algorithms involved processing of the peak magnitude sequence as well as extremal coding of the epoch duration sequence.

5.2 Extremal Coding Algorithms

In section 5.1 it was stated that two sets of algorithms were developed and that they differed in their treatment of the peak magnitude information. This section describes a) the encoding, decoding and synthesis of the epoch sequences and b) the encoding, decoding and synthesis of the peak magnitude sequence (hereafter referred to as magnitude sequence).

5.2.1 Epoch Encoding

From inspection of the epoch sequences in figures 5.2(a), 5.3(a) and 5.4(a) it was noted that, besides the existance of easily defined extreme, as seen in figure 5.5(a), conditions do occur where it may be argued that the extreme exists in one of two posit-

ions, or that both positions are true extrema. This condition can be observed in figure 5.4(a). In general, for speech waveforms, interpolation is employed to determine the precise position and amplitude of the extreme thus removing any uncertainty. Unfortunately, this technique is not applicable to the epoch sequences because the the data originates from a discrete source and is undefined between data points. For the purpose of extremal coding the extreme must exist in one of the two positions, but which ?.

A number of options were considered for resolving this dilemma. One possibility was to allocate an extra codeword which the receiver would interpret as meaning that two consecutive points of equal value formed the extreme. The receiver algorithm would then interpolate between the previous extreme and the first extreme position of the current extreme. The second extreme position would then be treated as the previous extreme thus retaining both extrema positions.

An alternative option involves the principle of a biased decision. When two samples 'contend' the extreme position, the encoder allocates it to either the first or second sample position under all conditions.

The inclusion of an extra codeword for signalling the occurrence of a dual sampled extreme would at least prevent the distortion of one epoch sequence sample. However, it was decided that, for an increase in algorithm complexity, no increase in quality would be perceived due to the masking effect of the distortions introduced by the surrounding epoch durations distorted by the encod-

and synthesis. It must also be noted that the use of a special codeword requires more data to be transmitted with little, if any, gain in quality therefore reducing the overall efficiency of the algorithm.

A decision on which position to choose for the option of biasing, or alternatively restricting the extreme position to the first or second sample position was an arbitrary choice. The extent of the distortion experienced by the rejected sample position was dependant upon the value of the next extreme if the first sample were chosen, or the previous extreme if the second sample were chosen. Clearly, the number of samples between extrema was also be a contributing factor. Since the biasing technique requires no extra codewords and enabled the algorithm to yield greater data reduction with little or no effect upon the overall speech quality, the final algorithm employed biasing to the first sample position. The choice of bias direction was arbitrary.

Figure 5.5(c) depicts the conditions where three samples of equal value form an extreme. Without added complexity, the algorithm would treat this condition in the same manner as that of the two sample extreme. However, this would result in excessive distortion of the samples between the current extreme and the next extreme detected. It is arguable that the second, or central, sample position should be treated as the extreme sample. If this were the case the neighbouring samples would be subjected to distortions similar to those experienced in the two sample extreme biasing. In one respect to choose the central position would be a suitable

solution, but, contingencies had to be made for the condition where four or more samples of equal value formed an extreme, albeit uncommon within the current coded speech file. If an even number of samples (greater than 2) formed an extreme, then ambiguity of extreme position existed since two samples contended the central sample position. Another disadvantage of choosing the central position would be the extent of epoch duration distortion. If, for example, an extreme was formed by five equal valued samples and the central sample was chosen as the extreme position, four of the five samples would be distorted. The degree to which these samples were distorted increased as the separation distance between sample position and chosen extreme sample position increased. Distortions of so many samples would become apparant since five undistorted epoch durations could represent upto 16ms. of synthesised speech.

Another factor considered was that of algorithm complexity and processing time. The greater the number of possible types of extreme the greater the algorithm complexity. Also the delay required before a decision as to which type of extreme had been detected would also increase. Clearly, a real-time implementation of such an algorithm would have an upper limit to the processing time available to be dedicated to such decisions. If too much processing time were spent on parameter analysis then the input sample buffer could rapidly fill resulting in data loss.

Rather than process 3, 4, 5, ... sample position extreme as separate events, where special measures were employed to handle even numbered sample events, it was decided that all extreme of 3 or more

sample positions would be processed identically.

If 3 successive sample positions had the same value then the first sample position was chosen as one extreme position and its value, with a code to represent the number of sample positions since the last extreme (distance measure) were transmitted. A count of the number of equal valued sample positions was then maintained. When a sample position which differed in value was detected the sample position prior to it was chosen as an extreme and its value and distance measure were transmitted. This technique is shown in figures 5.5(c) and (d) for extreme of 3 and 4 sample positions, respectively.

This technique has a number of advantages. The algorithm complexity is minimised. The extreme detection, processing and coding time is the same as that of the two sample extreme irrespective of the number of sample positions forming the extreme and the original extreme sample values remained undistorted. The algorithm complexity was minimised by the 'arbitrary' choice of biasing the two sampled extreme to the first sample.

While describing the real-time digital voice channel in chapter 2, silence signalling was also discussed. The codeword employed to signal silence was, in general, taken from the amplitude dictionary. However, there is no reason why a codeword from the tes-dictionary could not be employed. In such a situation the occurrence of extrema with at least three equal valued samples positions would be very frequent. Distortion of these samples could not be tolerated as they would be decoded by the TES decoder (not the extremal decoder)

as an ordinary epoch parameter. However, the technique described above would prevent distortions of this nature from occurring.

5.2.2 Peak Magnitude Encoding

In order to distinguish between the two algorithms developed they shall in future be referred to as EXTR1 and EXTR2, respectively. In section 5.2 it was indicated that the algorithms differed in their treatment of the magnitude information. In fact EXTR1 did not attempt to encode the magnitude information and transmited all values. If N samples occurred between extrema, the information transmitted was the extreme value and N+1 magnitude values. The distance measure was redundant and the (N+1)th magnitude was the value associated with the current extreme.

EXTR1 was developed in order to demonstrate (a) the quality of speech achievable without added distortions introduced by magnitude encoding, (b) that extremal coding affects the various speech sounds and (c) the average data required per epoch.

The algorithm EXTR2 utilised the extrema detection of the epoch duration sequence to signal coding of the magnitude sequence. Comparing the sequential epoch duration plots of figure 5.2(a), 5.3(a) and 5.4(a) with the corresponding sequential magnitude plots of figure 5.2(b), 5.3(b) and 5.4(b), it can be seen that when pronounced epoch sequence repetition occurs the extrema of the epoch and magnitude sequences coincide. This feature may be observed in greater detail by comparing figures 5.10(a) and 5.14(a) which are an

enlarged section of figures 5.4(a) and 5.4(b), respectively. During periods of erratic epoch sequence the magnitude values tend to be low levelled which hindered observations for similar epoch-magnitude relationships. To determine whether such a "phase" relationship also existed during erratic epoch sequences, enlarged sections of sequential epoch and magnitude plots were taken.

Comparisons of the enlarged epoch sequential plots of figures 5.7(a), 5.8(a) and 5.9(a) with their corresponding enlarged magnitude sequential plots of figures 5.11(a), 5.12(a) and 5.13(a), respectively, revealed, in general, for the short duration epochs (less than 0.75ms), the minima of the magnitude sequence coincides with those of the epochs. Unlike the periodic epoch sequences, from observations to date, no generalisation may be made concerning the magnitude sequences corresponding to the erratic epoch sequences.

It has previously been mentioned that the low level erratic epoch sequence occurs during unvoiced sounds. From the model of speech production and the discussion of vocoders in chapter 1 we have seen that the unvoiced sounds are generated by a random noise source. Singh [79] demonstrated that unvoiced sounds could be replaced by a noise source but "sharper synthesis" was achieved when spectrally shaped noise sources were employed.

The coding of the magnitude sequence utilised the "phase" relationship of the epoch and magnitude sequences. When an epoch extreme was detected, the corresponding epoch magnitude was transmitted with the extreme value and distance measure. At the receiver, the epoch and magnitude sequence were reconstructed in parallel.

Even though distortion of the magnitude sequence would be greatest within the unvoiced segments it was speculated that the distortions introduced into the synthesised speech would be more pronounced for the voiced rather than unvoiced segments. As reiterated above, unvoiced sounds can be represented by a random noise source and therefore it was conjectured that the erratic epoch segments would tolerate a greater degree of magnitude distortion than the periodic segments.

5.2.3 Decoding

Mathews [53] proposed two interpolation functions which preserved the extrema of the original signal and synthesised the waveform so that no discontinuities existed in that waveform or its derivative at its extreme. Once again such techniques are not applicable here due to the nature of the data. The epoch sequences between extrema are most accurately approximated by straight lines due to the rapid changes of epoch durations which exist. Also the epoch and magnitude sequences do not require an interpolation functions which ensures continuity at the extreme.

Straight line interpolation was therefore employed for the synthesis of the sequences betweem extrema and the extrema retained their original value. The interpolated values were rounded to the nearest integer.

EXTR2 also employed straight line interpolation for the synthesis of the magnitude sequence. However, it was discovered that

epoch elimination within the synthesised speech was possible during erratic epoch/low level magnitude sequences. Epoch elimination occurred when an epoch extreme was detected whose corresponding magnitude was zero valued and if the next extreme detected also had a zero valued magnitude, then on synthesis all magnitude values between these extrema would be zero valued. Figure 5.6 depicts the functioning of the algorithms EXTR1 and EXTR2 and includes an example of epoch elimination. Epoch elimination would resulted in periods of silence being inserted into the synthesised speech. The frequency of occurrence of epoch elimination was not a predictable parameter but the periods of silence introduced were, in general, would probably be of very short durations (0.3ms or less). Miller and Licklider [80] conducted investigations into interrupted speech which demonstrated that speech intelligibility would not be grossly affected by this form of distortion, although the quality would suffer.

In order to determine the average data rate and information per epoch, several counters and one dimensional arrays were incorporated into each algorithm. Their function was to establish the number of epoch/magnitudes processed, the number of codes transmitted and the frequency distribution of (a) the distance measure and (b) the epoch durations occurring as extreme. Neither algorithm set a limit upon the distance measure because observations upon the epoch duration sequences revealed that, unless the TES encoder utilised silence signalling, the distance measure between extrema rarely exceeded seven sample positions. In order to calculate data rates, a fictitious limit had to be imposed upon the distance measure.

5.3 Results

After processing the coded speech files (CSF) with either EXTR1 or EXTR2, sections of epoch and magnitude sequences which corresponded to the enlarged sections of the original sequence were plotted. This This enabled detailed comparisons of the original and processed sequences to be conducted and to establish the sources and extent of the distortions due to extremal coding. An algorithm for the synthesis of speech from the processed CSFs was then implemented. Informal comparisons of speech synthesised from both the processed and unprocessed CSFs were conducted to gain an insight into the way the distortions within the Tes domain were manifest in the time domain. Section 5.3.1 presents the discussion concerning the sequence comparisons and section 5.3.2 gives an informal subjective appraisal of the synthesised speech achieved using the CSFs output from EXTR1 and EXTR2. Section 5.3.3 presents the results of the power spectral density measurements.

5.3.1 Sequence Comparisons

Figures 5.2(a), 5.3(a) and 5.4(a) present the epoch duration sequencies input to the extremal coding algorithms and figures 5.2(b), 5.3(b) and 5.4(b) present the magnitude sequencies input to the extremal coding algorithms.

Table 5.1 defines the segments of the input epoch duration and magnitude sequencies for which enlarge sequential plots have been produced. For comparison purposes enlarged sequential plots were

also produced for the processed epoch duration and magnitude sequences. These plots are also defined in Table 5.1.

The initial comparisons of the original and processed epoch sequences gave the impression that extremal coding had been capable of reproducing the sequences without excessive distortion. Close comparison of figures 5.7(a) and 5.7(b) reveals that the majority of the sequences are identical for single sample position extreme and the deviations occurred where extrema were formed by two sample positions. The description of the encoding algorithm in the previous section had already highlighted the possible occurrence of such distortions at the extreme. Comparing figures 5.8(a) and 5.9(a) with their processed equivalents, figures 5.8(b) and 5.9(b), indicates that the majority of distortions again occurred when epochs of similar value were encountered. However, in these examples, the original epoch durations were, in general, less than 0.5ms.

Figure 5.10(a) gives the epoch sequence produced by a strong voiced sound, while figure 5.10(b) is the same sequence after extremal coding. These figures highlight two possible situations. The first exists when the algorithm yields a perfect reproduction of the original sequence (samples 100 to 125) and the second is that of the distortion caused by dual valued extreme, or a series of epoch durations of similar value occurring between extrema. These distortions result in an increase in the high frequency component when that section is employed for speech synthesis. The perceptual effect will be equivalent, atleast, to low level background noise.

From inspecting the reconstructed epoch sequences it was appar-

ent that time distortion would exist within the synthesised speech. The perceptual affect this has on the synthesised speech could not be predicted because the shorter epochs were distorted to a higher degree than the larger epochs when distorted by 'equal' amounts (see section 4.3.3).

The effect of the pseudo extremal coding of EXTR2 on the magnitude sequence was not unlike that previously described. The extent of the distortion depended on the distance measure. When a maximum distance measure of two occurred, the magnitude sequence was, in general, reproduced very accurately. This may be seen by comparing figures 5.11(a) and 5.11(b), inparticular sample numbers 115 to 145. The difference between these sections was minimal. The distortions became more pronounced as the distance measure increased or the "phase" relationship became non-existant. Referring back to figures 5.11(a) and 5.11(b), close inspection of samples 85, 95 and 110 reveal high attenuation of the original sequence. Since these samples originated from a voiced segment and originally were of both large epoch duration and magnitude then they were perceptually significant since distortions of voiced speech were more obvious than those of unvoiced speech.

Comparing figure 5.12(a) with 5.12(b) and 5.13(a) with 5.13 (b) there are a number of instances where excessive distortions had occurred. A large extreme value, at sample 105 of figure 5.12(a), has suffered severe attenuation. In figure 5.12(a), samples 127 to 133 and figure 5.13(a) samples 12 to 15, and 23 to 25 have been eliminated. The magnitude extreme at samples 135 to 137 and 148 to

150 of figure 5.12(a) have been merged. In this particular instance, as drastic as the distortions appear to be, the perceptual effect will probably be insignificant because this section corresponds to the transition between the /a/ and /t/ of "operator", where the /t/ is a voiced stop consonant and so the build up to the /t/ sound (which this section corresponded to) was low level noise. The occurrence of such a distortion else where within the sequence, would have had a dramatic effect upon the speech quality.

Finally, a very interesting comparison is that of the epoch duration probability distribution before and after extremal coding, presented in figures 5.14(c) and 5.14(d). The probability of an epoch duration of 0.05 or 0.1ms remains approximately the same, while that of epoch durations of 0.15 to 0.3ms, which were the most probable originally, has been reduced significantly. The most probable epoch duration within the extremal coded sequence was 0.7ms in duration. This was a result of the interpolation. As will be shown later, the majority of the extrema were either the short epoch durations of unvoiced sound (.LT. 0.5ms) or durations greater than 1.0ms (during voiced sounds). The linear interpolation between extrema caused the increase in intermediate values of epoch duration. For epoch durations greater than 1.5ms, the probability distribution remained very similar to that of the original. The shifting of the epoch duration probability distribution may manifest as an increase in the mid-band frequency components. Table 5.1 summaries the figures discussed within this section.

5.3.2 Informal Subjective Appraisal

The speech synthesised employing the unprocessed coded speech files (CSF) (the data input to the transform algorithms) was of high quality and intelligibility. The description of distortions in the speech synthesised utilising processed CSFs are therefore relative to that produced employing the unprocessed CSFs.

The CSFs output from the transform algorithms were utilised as the input files for the speech synthesis algorithm described in chapter 2. The synthesised speech output was bandlimited (300 to 3400Hz).

Informal listening of the speech synthesised from the CSF output of the epoch duration extremal coding algorithm (EXTR1) indicated that a high degree of intelligibility had been retained. The quality varied over the test utterance and in the most severe cases sounded grainy and rasping. The speakers were still recognisable but it was the utterances of the male speaker which had incurred the majority of the distortions. The female utterances, in general, had a very crisp sound to them.

The male utterance of "I'd like to make" had a very drawled quality, while the utterance of "No I said Balam in England" took on a more irrate tone than that of the original speech. These effects were believed to be a result of the time distortion introduced by the linear interpolation. The slow drawling sound being a result of an increase in the total time of the utterance and the irrate characteristic due to a decrease in time.

The majority of the nasal and stop consonant sounds were very distinct. The /t/ of "what" was distorted but the extent of which was not sufficient to affect the passage. The grainy rasping sounds mentioned previously were prominent during voiced sounds. This was understandable since a number of epoch durations during the voiced sounds had suffered considerable distortion.

As expected, the quality of the speech produced by the synthesised of speech employing the output of EXTR2 was less than that of EXTR1. The male speaker sounded muffled and raspy. The female spoken sections were again found to be more robust against distortion. The female utterance "did you say" was very clear with the ending of "say" beginning to become raspy. The nasal sounds /m/ of "Balam" by the male speaker was totally obscured. Again the female nasal sounds were more distinct even though they had suffered some distortion.

However, although the quality of the female spoken sections were believed to be better than that of the male, this was only true when one compensated for the interrupt nature of one female utterance. The utterance "What part of England is that" was broken up by several short silence intervals due to the occurrence of magnitude elimination in the Tes domain. This, combined with the poor reproduction of the plosive sounds, gave the speech a hesitant quality, yet intelligibility and speaker identification were not unduly affected. Once again time distortion was very evident.

5.3.3 Power Spectral Density Measurements

Extremal coding of the epoch duration sequence and pseudo-extremal coding of the magnitude sequence are non-linear, signal dependant techniques. The effect of these coding techniques, combined with the Tes transformations, had upon the power spectra of the speech cannot be predicted. However, from figures 5.14(c) and (d) it was observed that the probability of epochs in the region of 0.4ms (8 samples) and 1.4ms (28 samples) had been significantly increased by the interpolation process. It was therefore conjectured that the increased probability would result in considerable mid-band changes within the power spectra. However, in general, comparisons of the power spectra have not revealed any significant trends which could be attributed to this.

The speech segment and Power Spectral Density Plot (PSDP) of figures 5.15(a) and (b) correspond to a transition from a stop consonant to a voiced sound. Comparing these with figures 5.16(a) and (b), which are the equivalent segment after the epoch durations had been processed by EXTR1, the spectra are seen to differ considerably at frequencies greater than 600Hz. In the Tes domain, some some voiced sounds do not exhibit high periodicity in the epoch duration sequence. The power spectrum for such sequences after processing by EXTR1 differed from the original spectrum but followed its general trend. Major departures were observed for the epoch duration sequences relating to the unvoiced and transitional sounds. During such sounds the spectra were, in some instances, vastly different to that of the original yet, from the previous discussion on the

informal listening, the unvoiced sounds appeared to be more robust to the distortions introduced by the extremal coding than the voiced sounds.

The variations within the power spectrum were due to the distortion introduced by the linear interpolation synthesis of the epoch duration sequence. These distortions were similar to those introduced by the smoothing algorithms investigated in chapter 4. The extent of the distortions were dependant upon the relative position of the epoch duration sample position to the extreme, between which the interpolation was being conducted, together with the accuracy of the interpolation of the sequence compared with the original. Although individually the distortions were not very significant, the overall effect was very apparant on inspection of the synthesisd speech and Power Spectral Density Plots, PSDP.

The effects of magnitude elimination in the Tes-domain, as described in sections 5.2.3 and 5.3.2, can be seen by comparing figure 5.15(a) (the original waveform) with figure 5.17(a) (the waveform synthesised using the output of EXTR2). A segment which originally corresponded to 2.1ms. of silence became extended to 4.4 ms.

The pseudo-extremal coding of the magnitude sequence caused only locally small variations of the power spectra, when compared with the power spectra for EXTR1. However, the differences that existed were clearly significant because the quality and intelligibility of the speech produced by synthesising EXTR2's output was informally judged to be inferior to that for EXTR1.

5.4 Data Reductions

The output of counters embeded into the algorithms indicated that 13,824 epoch durations were processed and 6,659 codes were transmitted. On first inspection a compression ratio of 2:1 had thus been achieved. However, a code consists of the epoch duration and a distance measure. The transmitted epoch durations were of the same accuracy as those processed (6 bits), thereby eliminating quantisation errors and distortions during the encoding and decoding, and the distance measure was not restricted. Therefore, to calculate the average number of bits per epoch the distance measure distribution had to be inspected. Using this, the average number of bits per epoch were calculated for specific values of restricted distance measurements (r.d.m).

The probability distribution of the distance measure is given in figure 5.18(a). As expected, the shorter and zero distance measures were the more probable and distance measures greater than 9 were not encountered.

To implement r.d.m extremal coding it was envisaged that when the maximum distance occurred and no extreme has been detected, the last sample would be designated an extreme. The relevant codes would be transmitted and the processing would continue until an extreme was detected or the maximum distance measure was once more reached. If the distance measure were restricted to 3 (thus requiring two bits) then a distance measure of five would be transformed into two sets of codes. That is, one set of codes for the maximum distance

of 3 plus the value of the epoch duration at that point and another for a distance measure of 2 plus the extreme value.

In order to calculate the average data per epoch two case of r.d.m were investigated. These were:

(a) r.d.m of 3 (2 bit representation)

(b) r.d.m of 7 (3 bit representation)

If an r.d.m of 3 were implemented the number of codes transmitted would have been 7,594, each consisting of 2 bits for distance and 6 bits for extreme representation. Therefore, a total of 60,753 bit would be required to represent 13,824 epoch durations resulting in an average representation of 4.4 bits per epoch.

An r.d.m of 7 would have resulted in 6,961 codes being transmitted, each consisting of 3 bits for distance and 6 bits for extreme representation. A total of 62,649 bits would therefore be required yielding an average representation of 4.5 bits per epoch.

For an r.d.m of 3 or 7 an average data compression ratio of 1.36:1 and 1.33:1, respectively, may be achieved using r.d.m extremal coding of the epoch duration sequence.

These results apply only to compression of the epoch duration sequence. The data compression achieved by EXTR2 would be greater since the magnitide sequence is also encoded. For an r.d.m of 3 and the magnitude values transmitted with the same accuracy as within the coded speech files (9 bits) a total of 17 bits per exteme would be required. This was equivalent to 129,098 bits for the represent-

ation of 13,824 epochs, an average representation of 9.34 bits per epoch instead of 15 bits (6 bits epoch duration and 9 bits magnitude). For an r.d.m of seven 18 bits per code transmitted would be required, a total of 125,298 bits for representing 13,824 epochs, an average of 9.06 bits per epoch.

Therefore, with an r.d.m of 3 or 7 an average data compression ratio of 1.61:1 and 1.65:1, respectively, may be achieved using extremal coding with r.d.m of the epoch duration sequence. Table 5.2 summaries the results presented above.

5.4.1 Parameter Coding

In section 5.4 it was shown that EXTR2 yielded better data compression than EXTR1. However, the quality of speech from EXTR1 was subjectively better than that of EXTR2. In order to further increase the data compression ratio the possibilities of transmitting different parameters to those currently employed or non-linear quantisation of the extrema values was inspected.

Given two extrema, E_1 and E_2, and the number of sample positions between extrema, N, the interpolation value, I_v, can be calculated. The interpolation value, which may be calculated using equation (5.1)

$$I_v = \frac{(E_1 - E_2)}{N + 1} \qquad (5.1)$$

is the increment by which E_1 is adjusted to yield the first interpolated sample between E_1 and E_2. Figure 5.18(b) gives the First

Order probability distribution of the absolute values of the interpolation value. From the distribution, we see that the probability of an interpolation value greater than 1.6ms is less than 0.0025. It was therefore suggested that instead of transmitting the extreme value and distance code, the interpolation value could be calculated to 5 bit accuracy (32 level) within the transmitter and transmitted with the distance measure. The receiver would assume alternating polarity of I_v, except for when $I_v = 0$.

On first inspection this approach appeared to be a plausible technique. However, when r.d.m's are imposed, some polarity difficulties occur. An example where an r.d.m of 3 has been applied, and the distance measure exceeds this value, is depicted in figure 5.19 (a). When the distance measure equals the maximum value, the corresponding sample position is chosen as an extreme. The interpolation value is calculated and coded along with the distance measure and transmitted. At the receiver, the codes are decoded and the interpolation values are given alternating polarity. However, the polarity of the interpolated values did not alter when the distance between extrema was greater than the r.d.m but, the epoch duration sequence would be reconstructed as if the polarity had changed. Thus the sequence is severely distorted. It may not be assumed that when a value for the distance measure which equals the r.d.m, is transmitted, the next interpolated value and distance count received are a continuation of the previous distance measure because distance measures equal to the r.d.m have a finite probability of occurrence.

In order to overcome the problems associated with the intro-

duction of an r.d.m, the polarity of the interpolated value must be transmitted. This means that, for increased complexity in the transmitter algorithm and possibly no improvement in speech quality, more data must be transmitted resulting in a reduction of the data compression so far achieved.

A further technique suggested was to implement 4 or 5 bit non-linear quantisation of the epoch duration extrema values. Figure 5.19(b) illustrates the probability distribution of the epoch durations occurring as extrema for the unrestricted distance measure. However, a major problem perceived with this technique is one of changing distributions. The probability distribution is dependant upon the utterance and is therefore quasi-stationary. To impose non-linear quantisation would be sub-optimal resulting in degradation of certain speech sounds. Further degradation of the speech may cause severe loss of intelligibility, particularly for EXTR2.

5.5 Conclusions

These investigations have demonstrated that the restricted distance measure (r.d.m) extremal coding of the epoch durations (EXTR1) can yield data compression ratios of 1.36:1 (with an r.d.m of 3) and 1.33:1 (with an r.d.m of 7). The synthesised speech was informally judged to have retained a high degree of intelligibility but varied in quality. The characteristics of speakers, in particular the male utterances, were found to have altered.

When the peak magnitude sequences were pseudo-extremally coded

(EXTR2) overall data compression ratios of 1.61:1 (with an r.d.m of 3) and 1.65:1 (with an r.d.m of 7) were achieved. However, a feature of this form of coding was the "zeroing" of epoch peak amplitudes which manifested as silence in the synthesised speech. The overall quality was judged to be inferior to that synthesised from the output of EXTR1.

		Enlarged Plots	
Section	Samples	Original Epoch Durations	Processed by EXTR1 and EXTR2
5.2(a),(b)	65 - 145	5.7(a)	5.7(b)
5.3(a),(b)	100 - 170	5.8(a)	5.8(b)
5.4(a),(b)	0 - 50	5.9(a)	5.9(b)
5.4(a),(b)	70 - 150	5.10(a)	5.10(b)

		Enlarged Plots	
Section	Samples	Original Peak Magnitudes	Processed By EXTR2
5.2(a),(b)	65 - 145	5.11(a)	5.11(b)
5.3(a),(b)	100 - 170	5.12(a)	5.12(b)
5.4(a),(b)	0 - 50	5.13(a)	5.13(b)
5.4(a),(b)	70 - 150	5.14(a)	5.14(b)

Table 5.1 : Cross reference of Figures and summary of epoch duration and magnitude sequencies processed.

	Restricted Distance Measure (r.d.m), (bits)			
	2		3	
Process	Bits/epoch	Compression	Bits/epoch	Compression
EXTR1	4.4	1.36:1	4.5	1.33:1
EXTR2	9.34	1.61:1	9.06	1.65:1

Table 5.2 : Summary of Data Compression achieved.

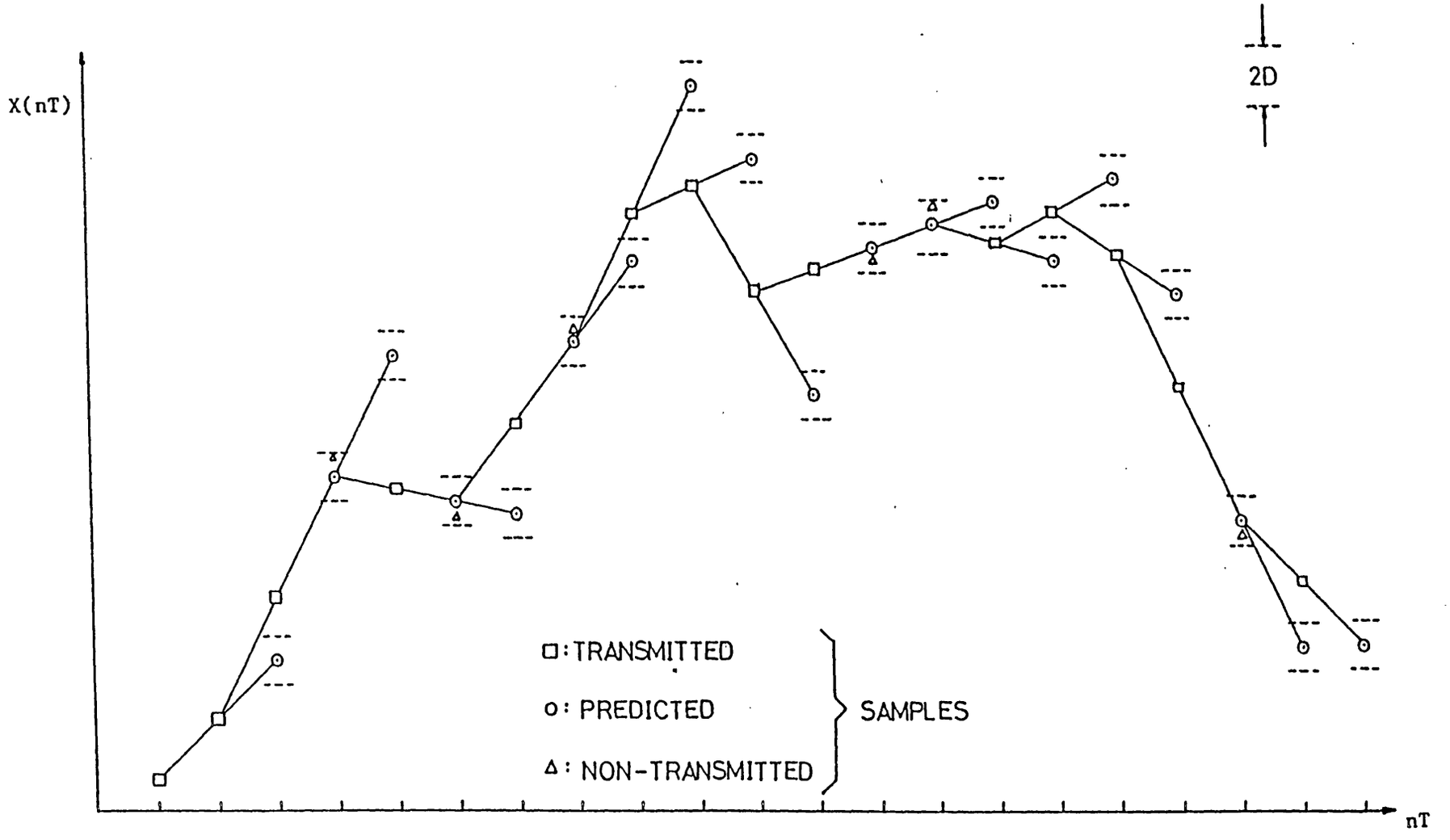


Figure 5.1 : Principle of First-Order Predictor (F.O.P) floating aperture.

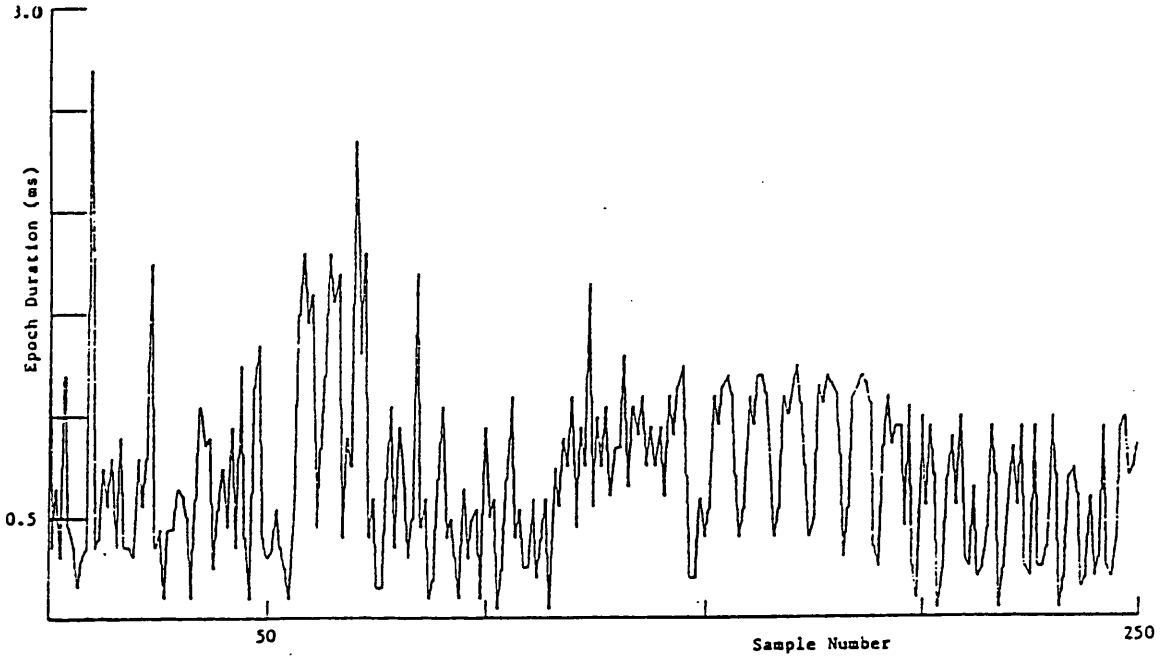
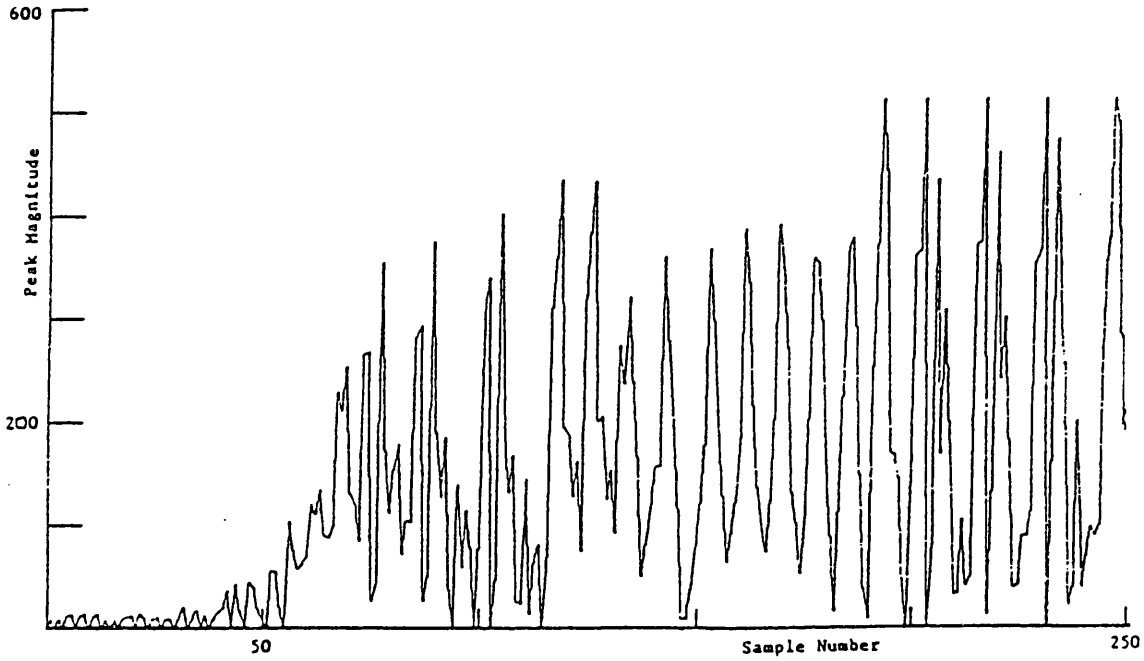


Figure 5.2 : (a) Sequence of epoch durations output from Al-Doubooni's Tes coding algorithm.

(b) Corresponding sequence of epoch peak magnitudes.



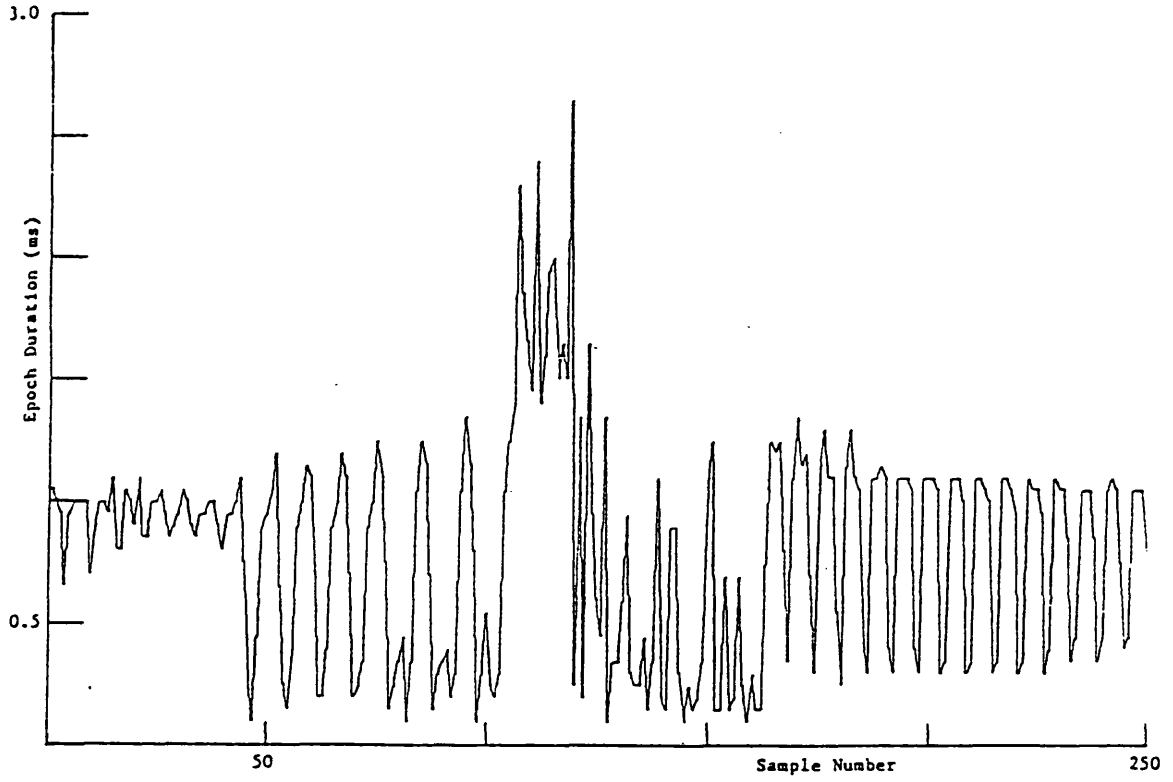
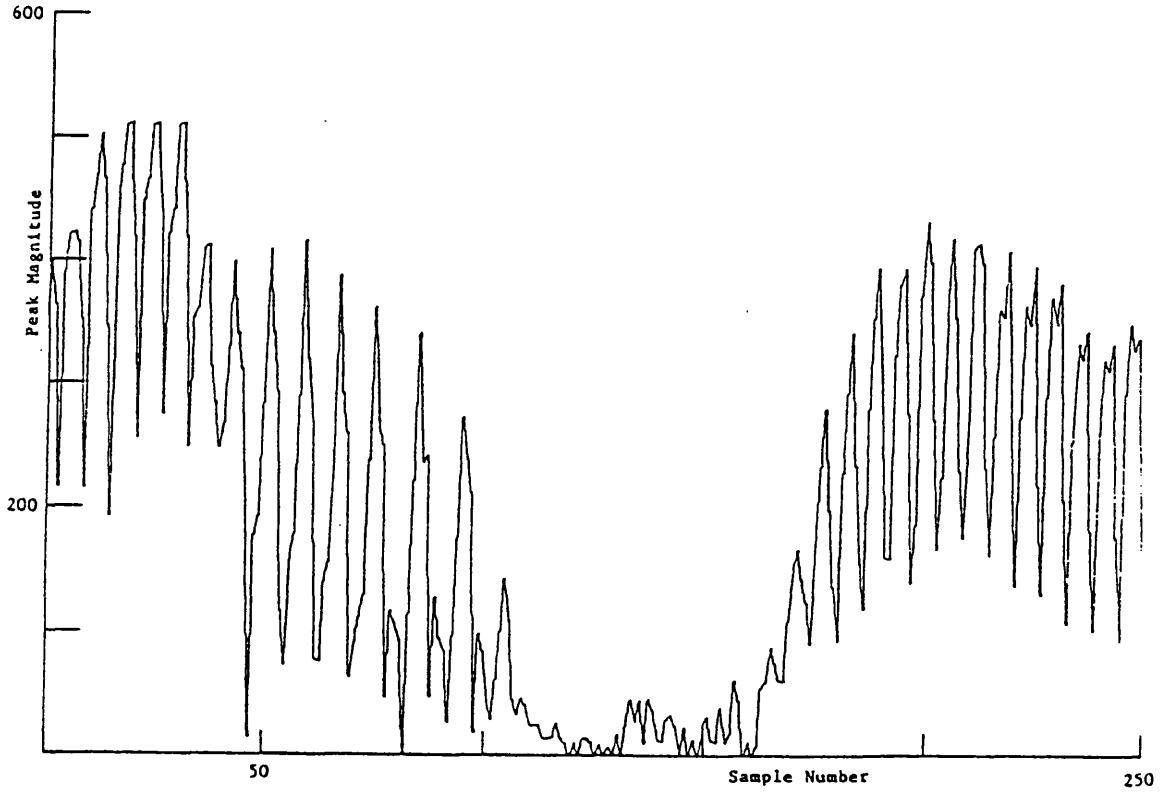


Figure 5.3 : (a) Sequence of epoch durations output from Al-Doubooni's Tes coding algorithm.

(b) Corresponding sequence of epoch peak magnitudes.



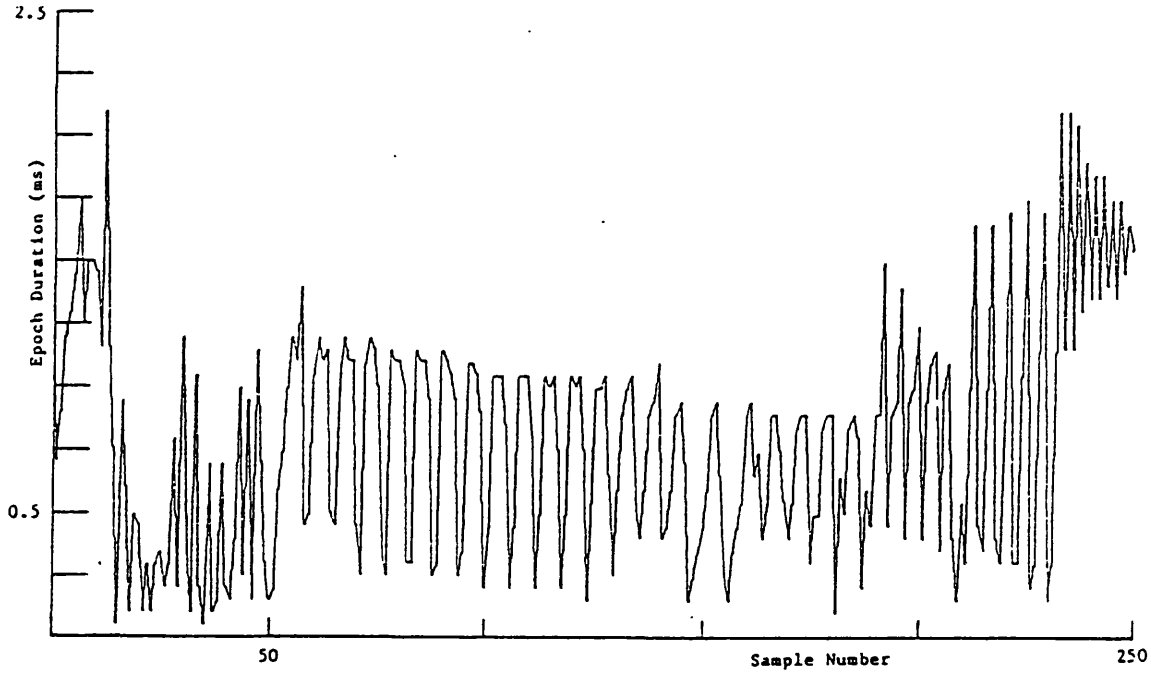
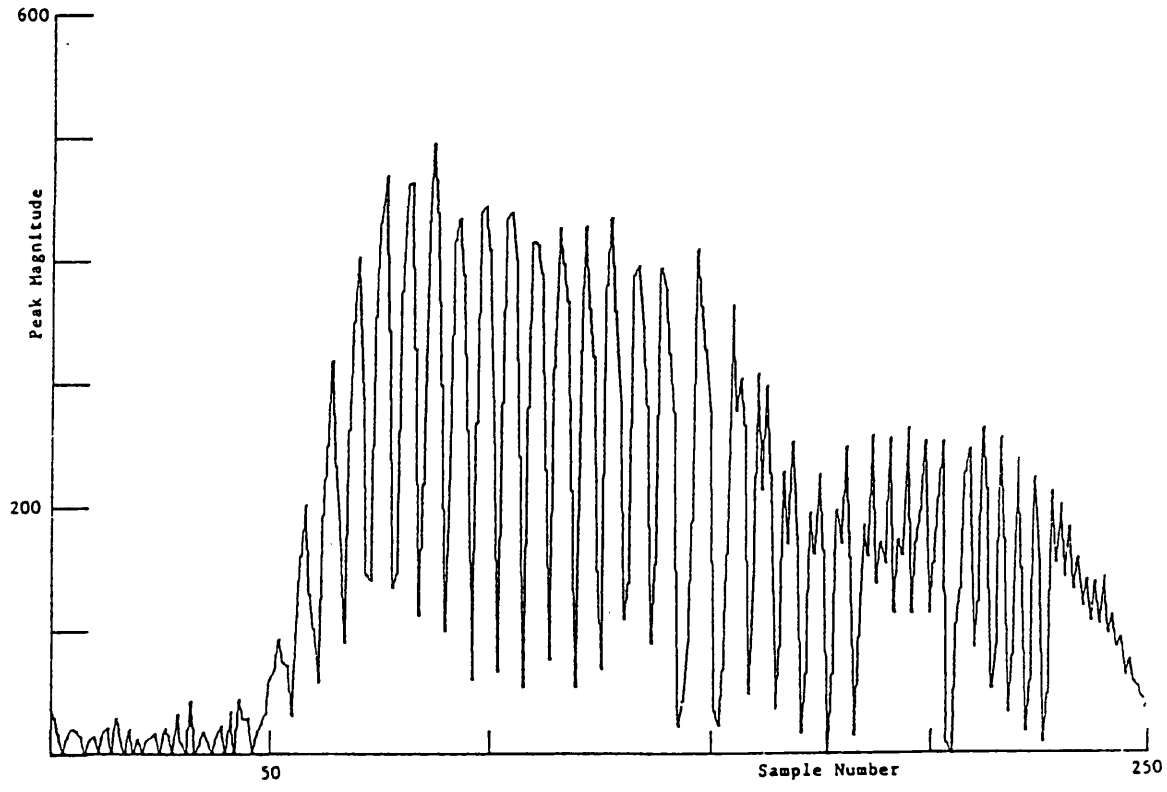


Figure 5.4 : (a) Sequence of epoch durations output from Al-Doubooni's Tes coding algorithm.

(b) Corresponding sequence of epoch peak magnitudes.



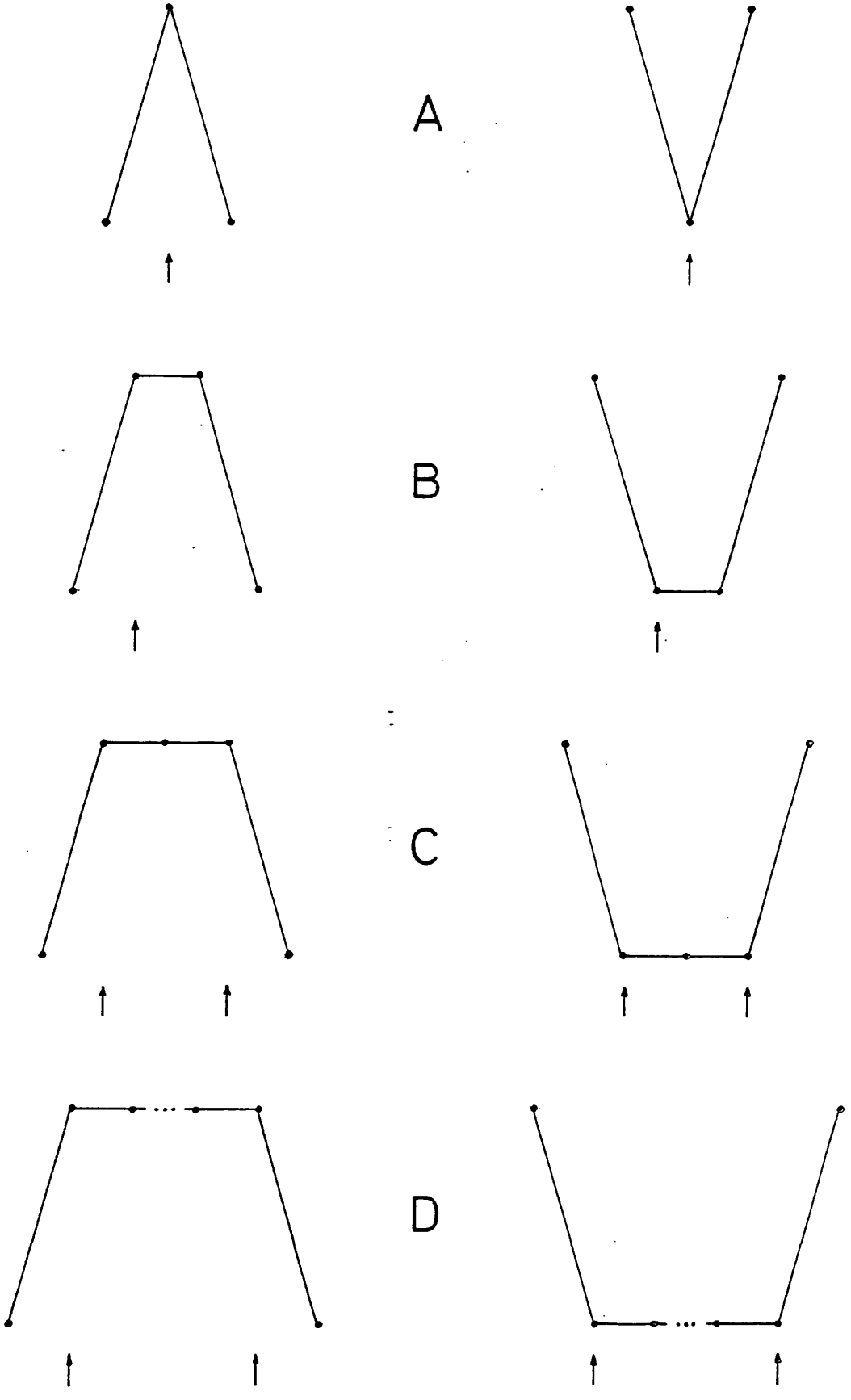


Figure 5.5 : Possible Epoch sequences with the extreme detected by EXTR1 and EXTR2 indicated.

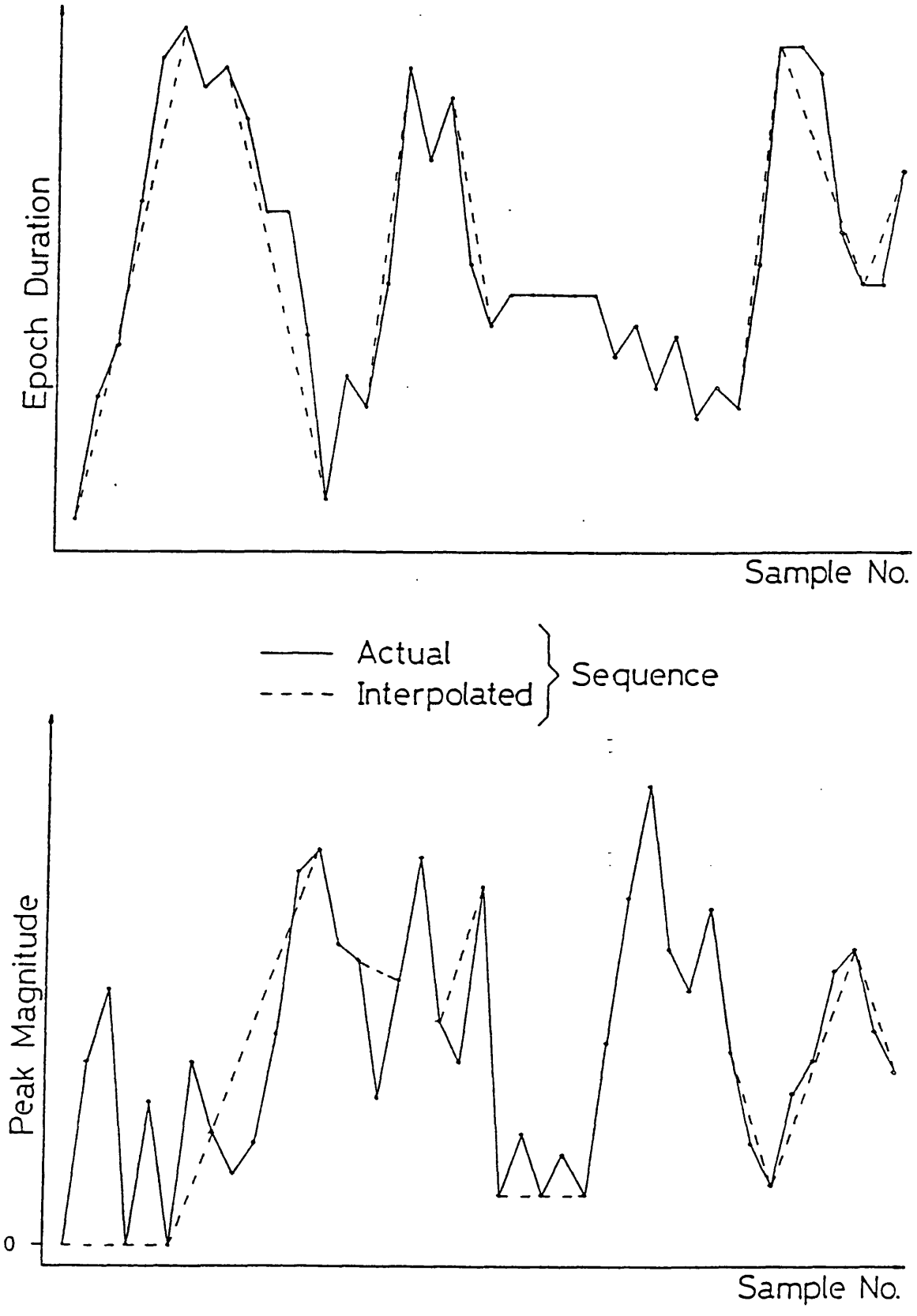


Figure 5.6 : Functional representation of the algorithms EXTR1 and EXTR2.

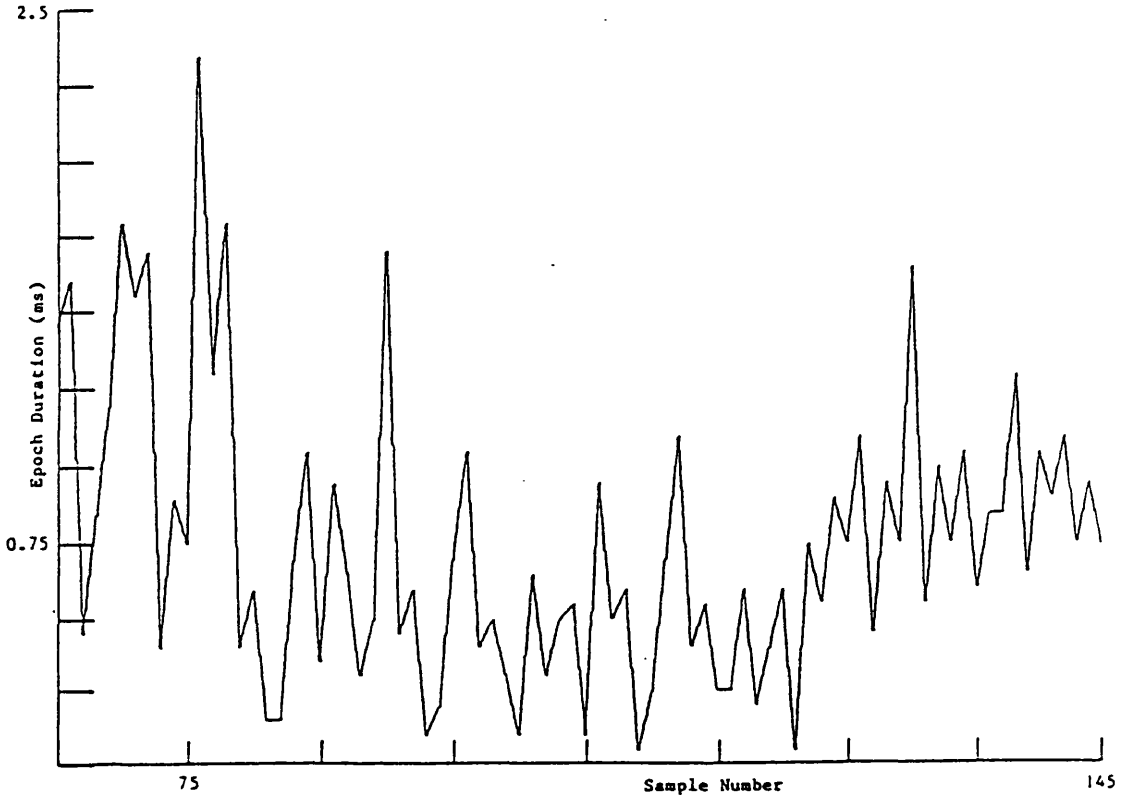
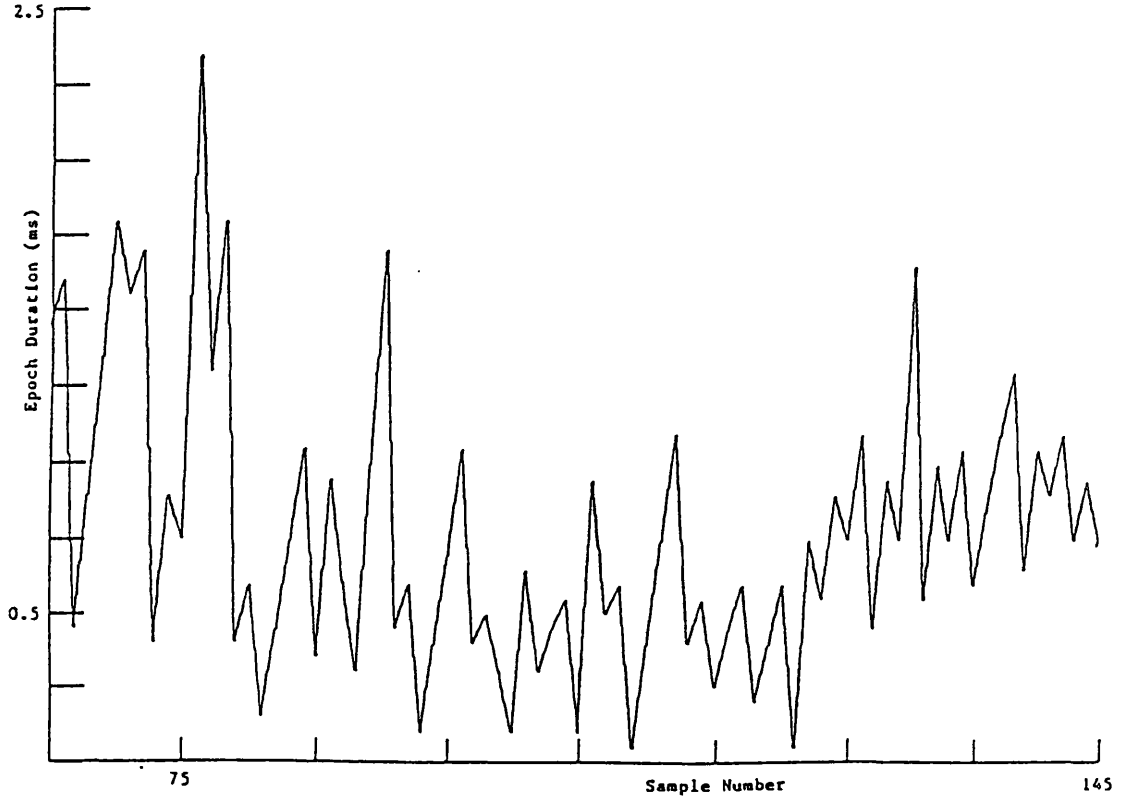


Figure 5.7 : (a) Enlarged segment of Figure 5.2(a).

(b) Equivalent segment to Figure 5.7(a) after processing by EXTR1.



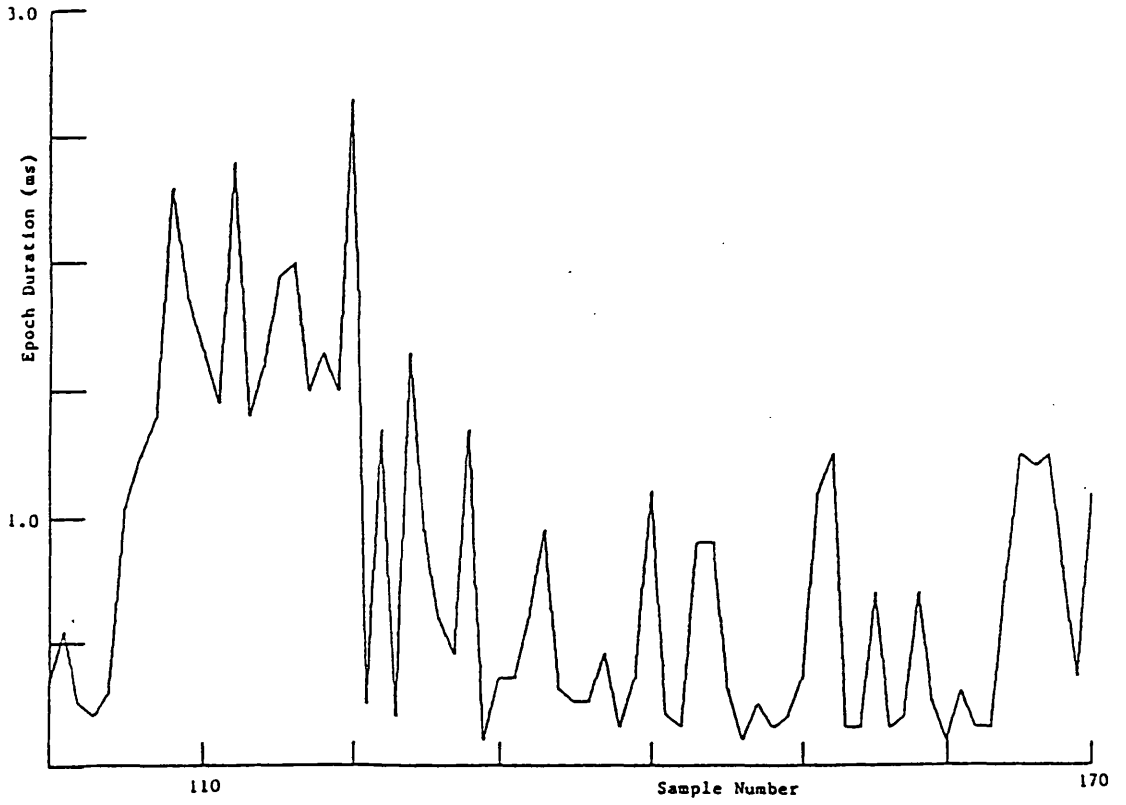
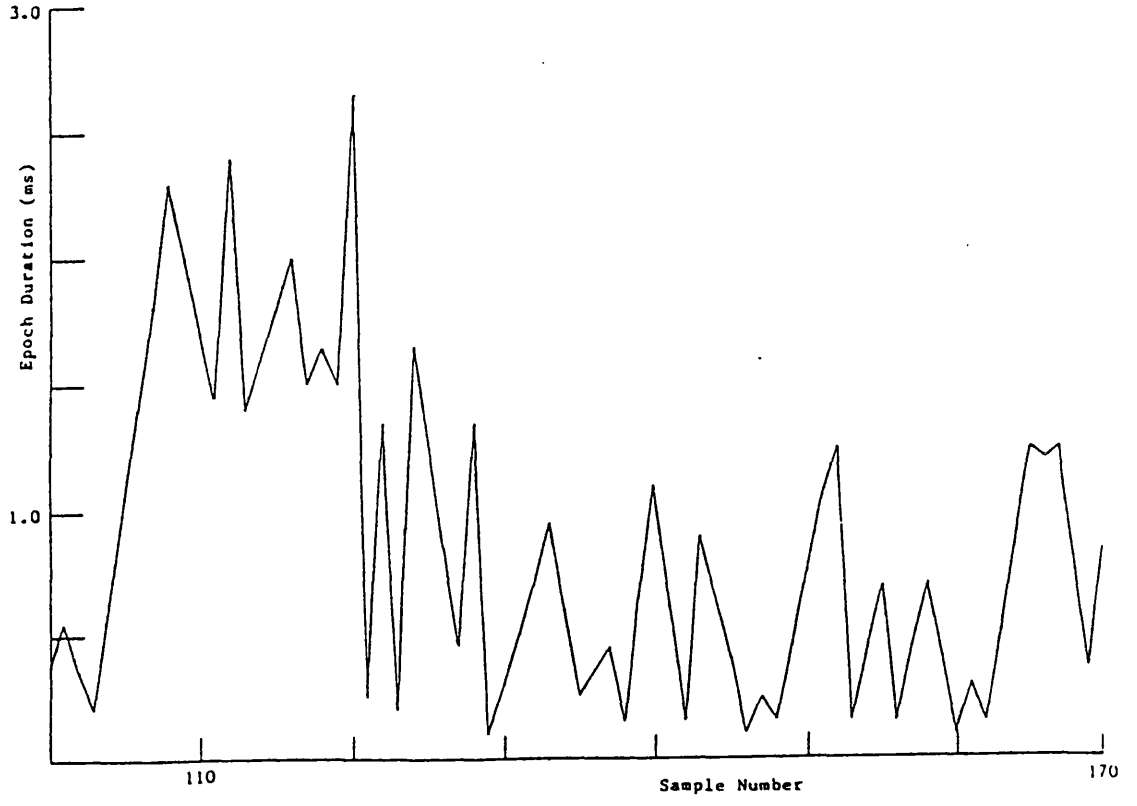


Figure 5.8 : (a) Enlarged segment of Figure 5.3(a).

(b) Equivalent segment to Figure 5.8(a) after processing by EXTR1.



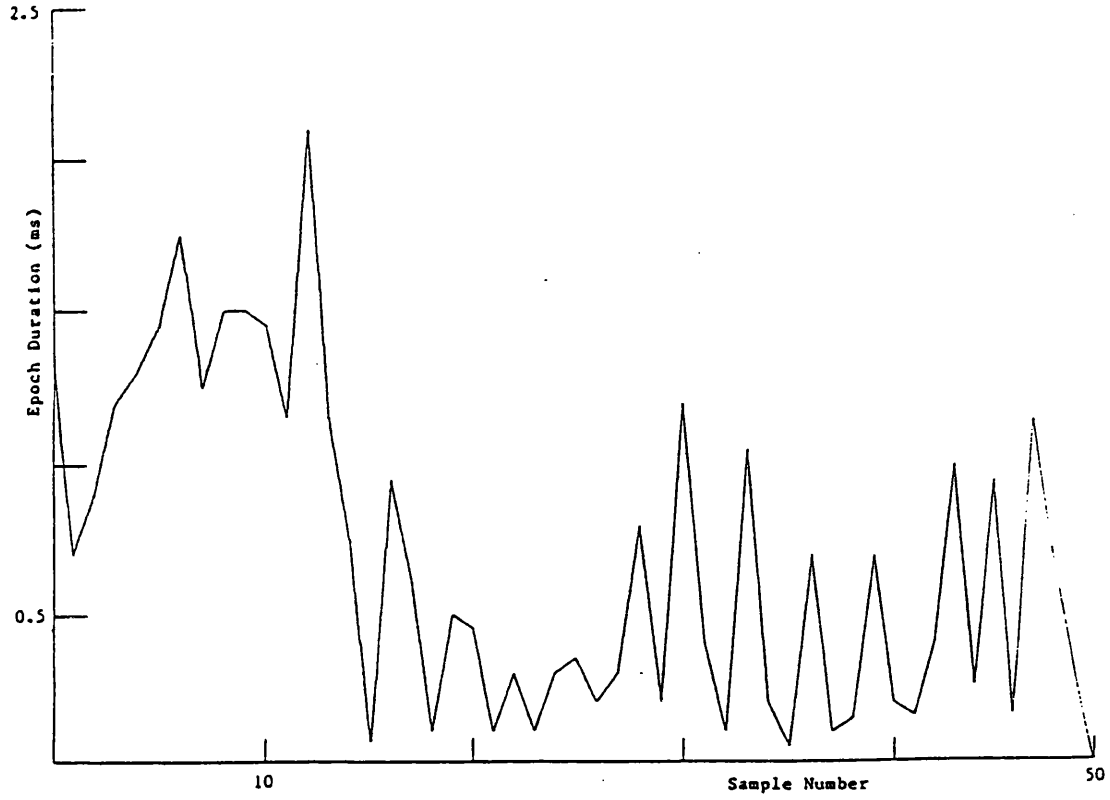
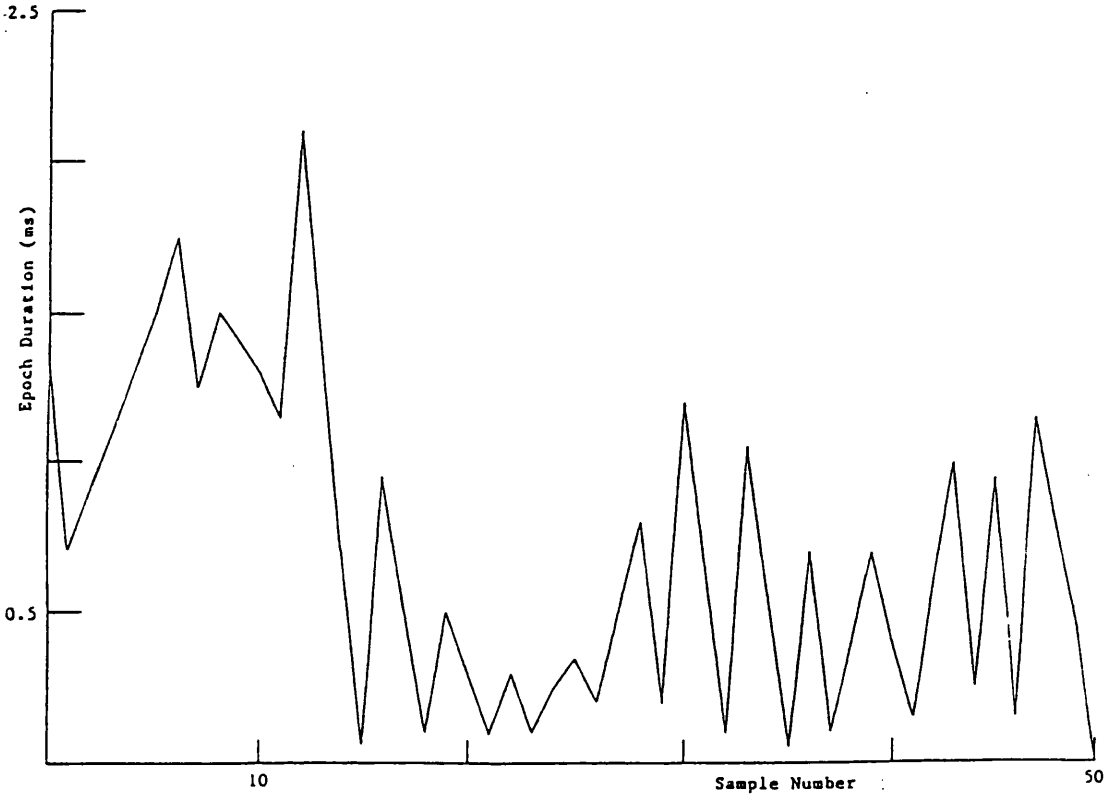


Figure 5.9 : (a) Enlarged segment of Figure 5.4(a).

(b) Equivalent segment to Figure 5.9(a) after processing by EXTR1.



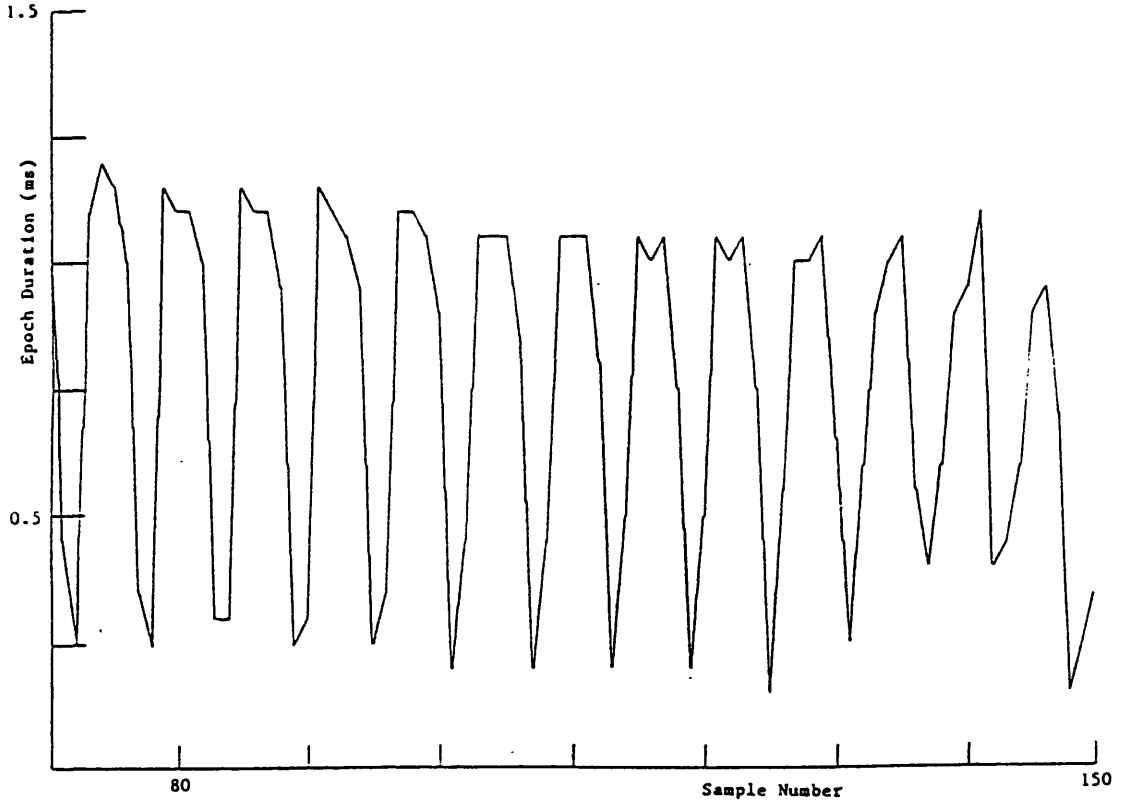
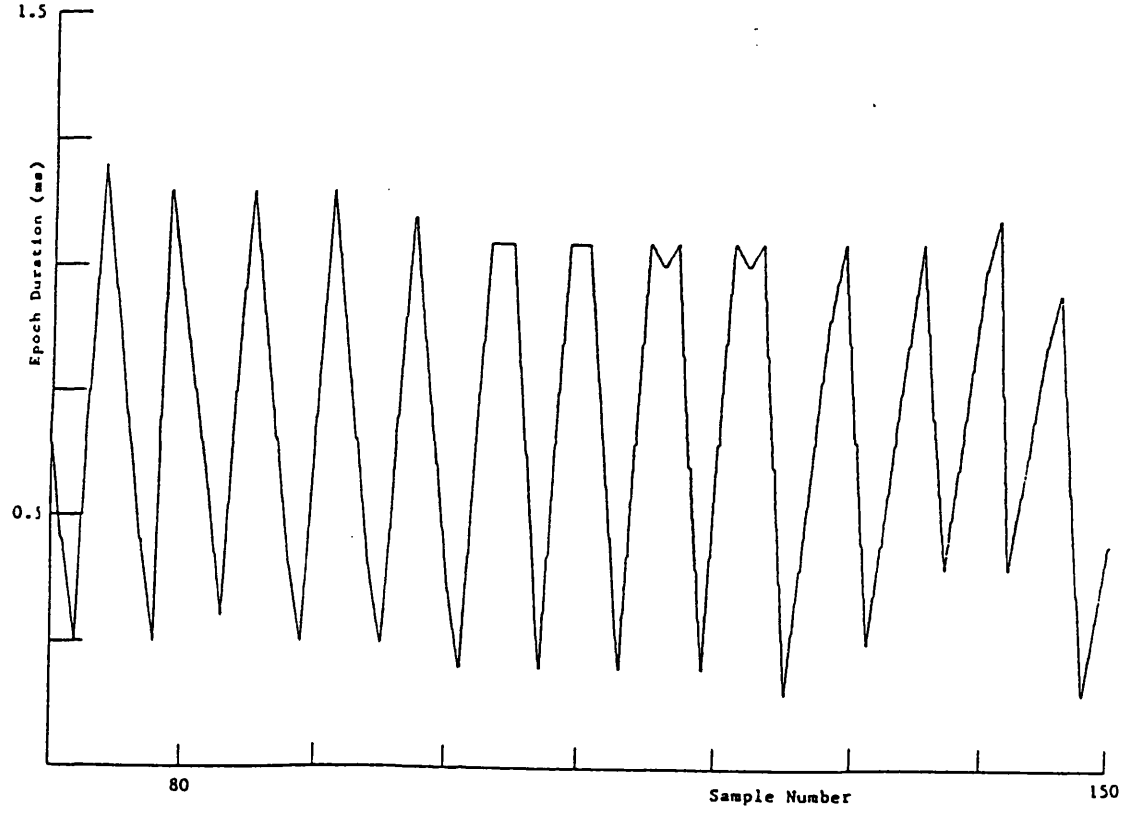


Figure 5.10 : (a) Enlarged segment of Figure 5.5(a).

(b) Equivalent segment to Figure 5.10(a) after processing by EXTR1.



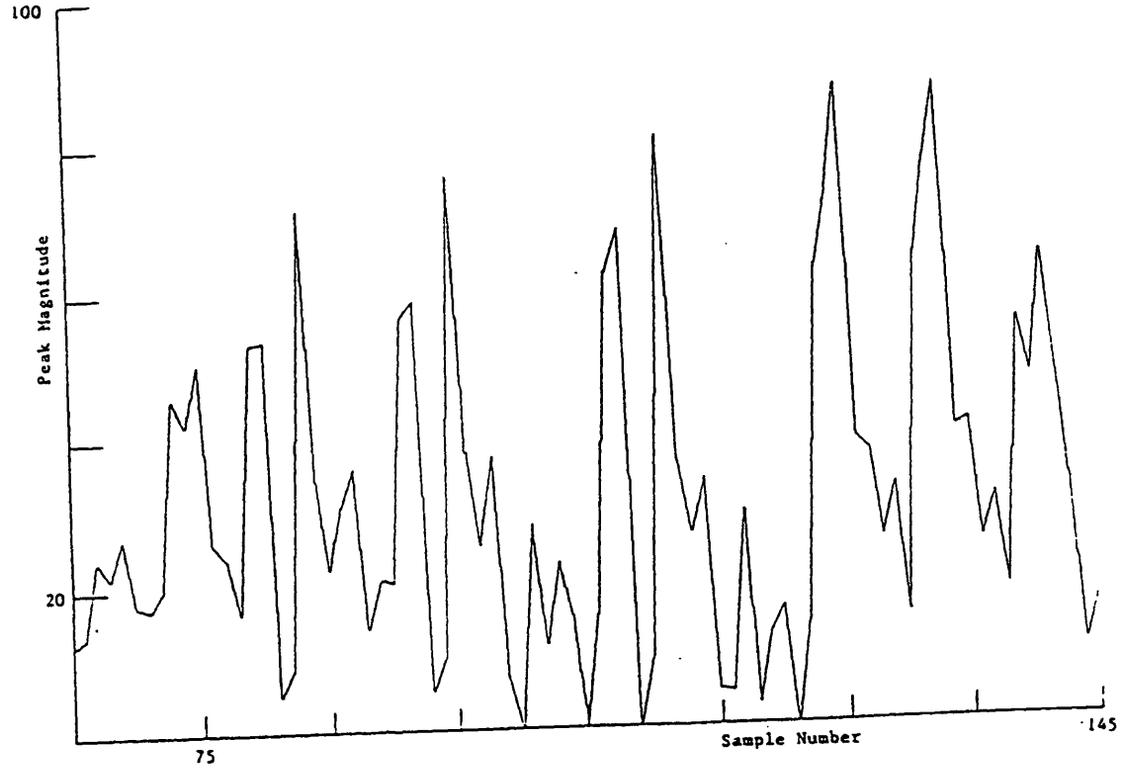
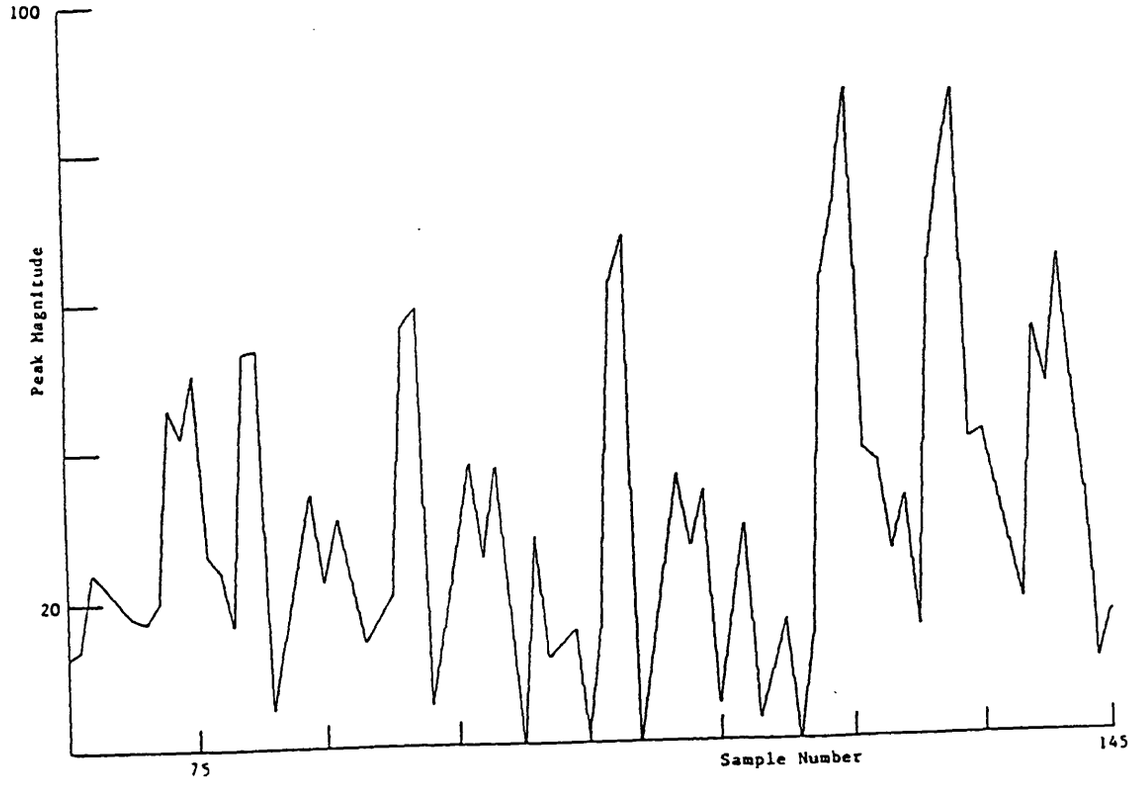


Figure 5.11 : (a) Enlarged segment of Figure 5.2(b).

(b) Equivalent segment to Figure 5.11(b) after processing by EXTR2.



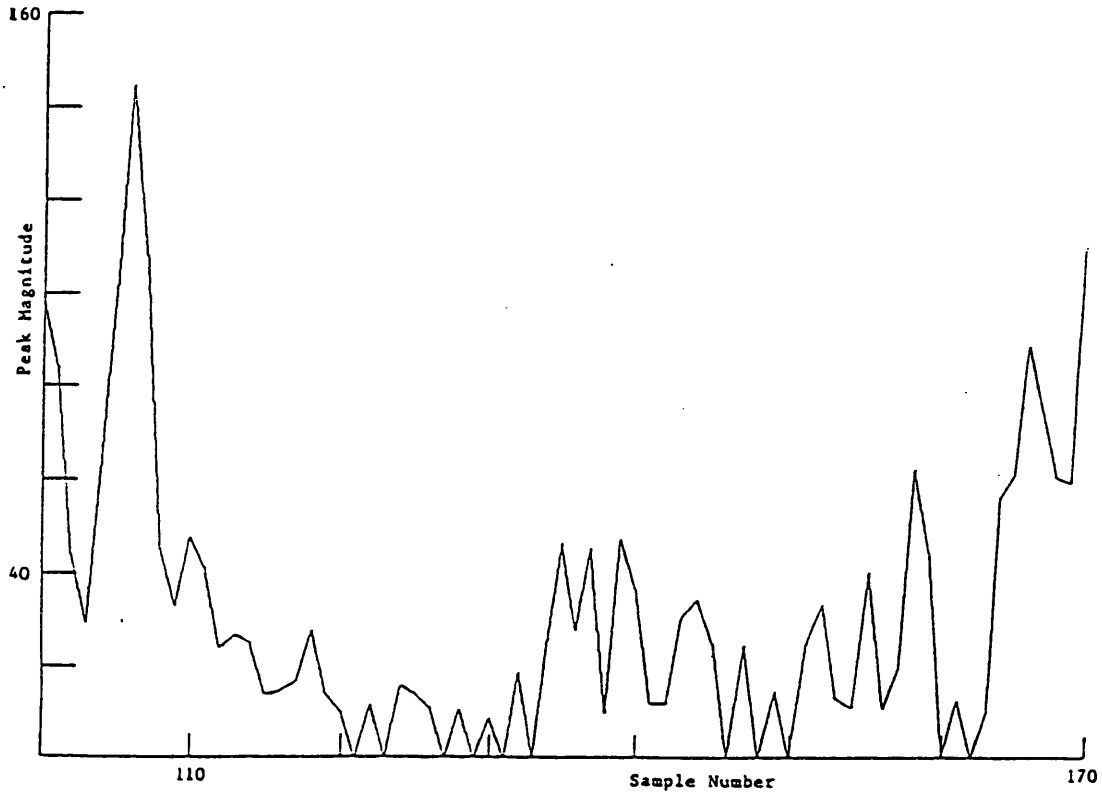
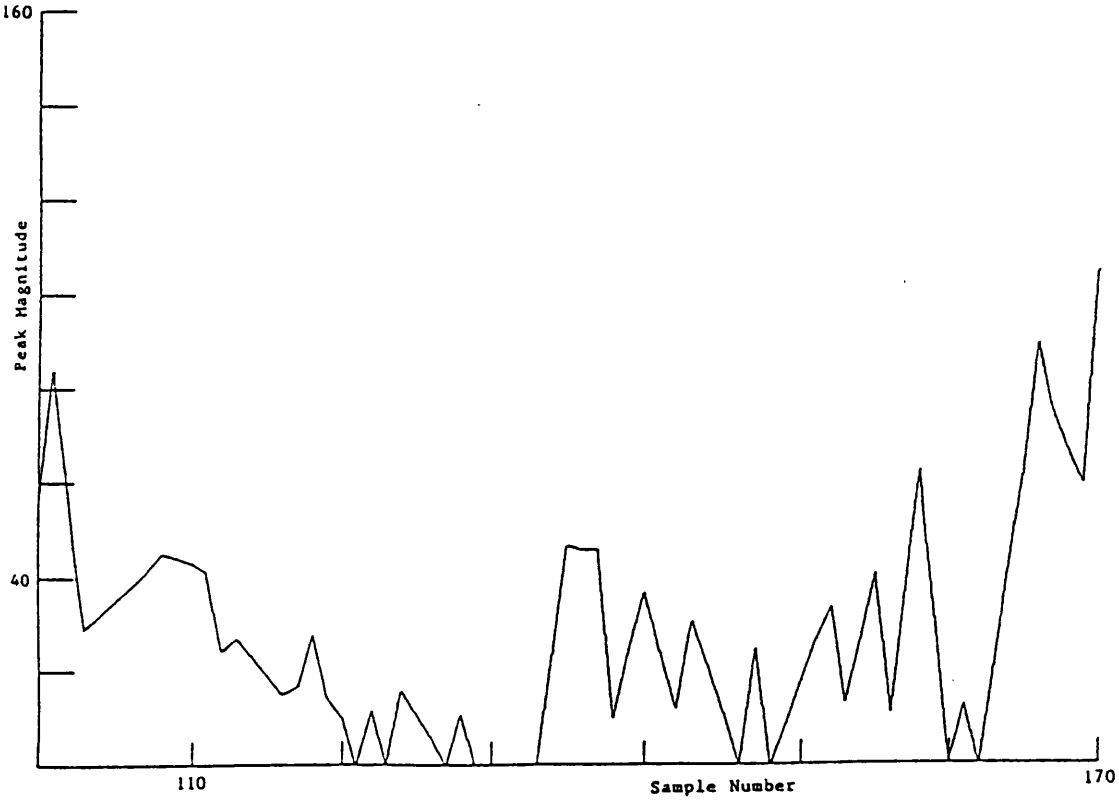


Figure 5.12 : (a) Enlarged segment of Figure 5.3(b).

(b) Equivalent segment to Figure 5.12(b) after processing by EXTR2.



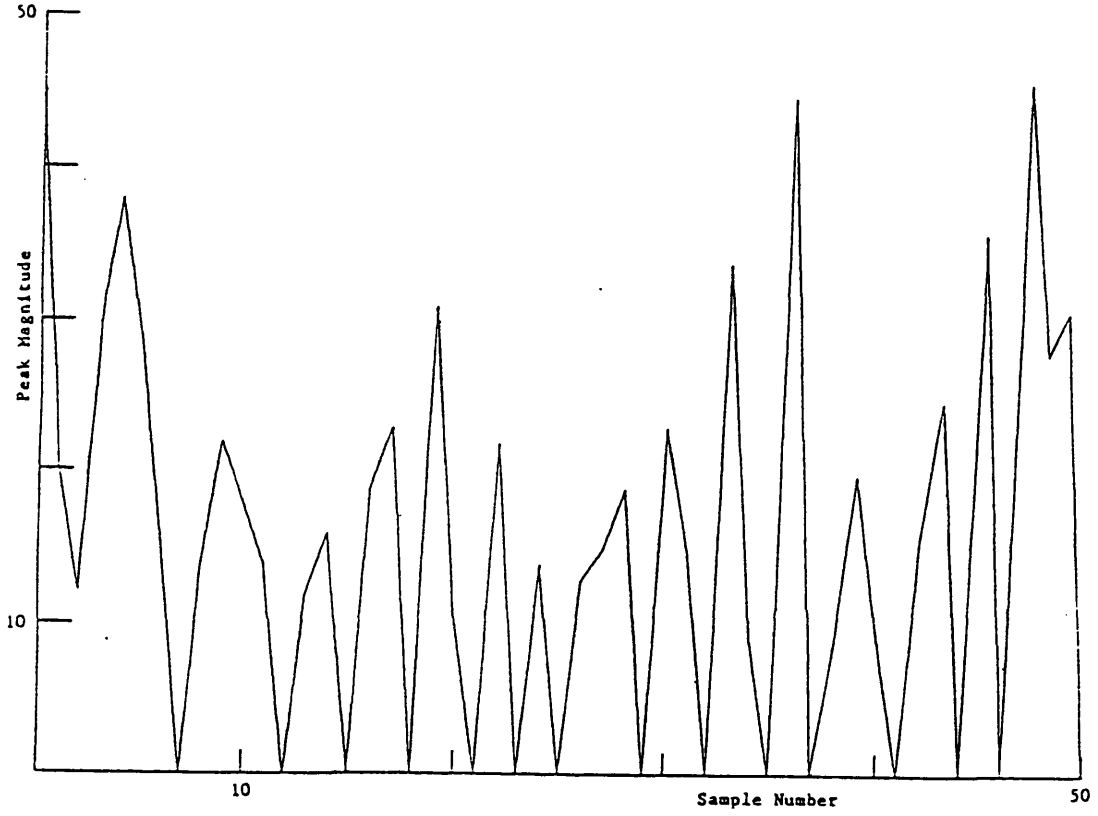
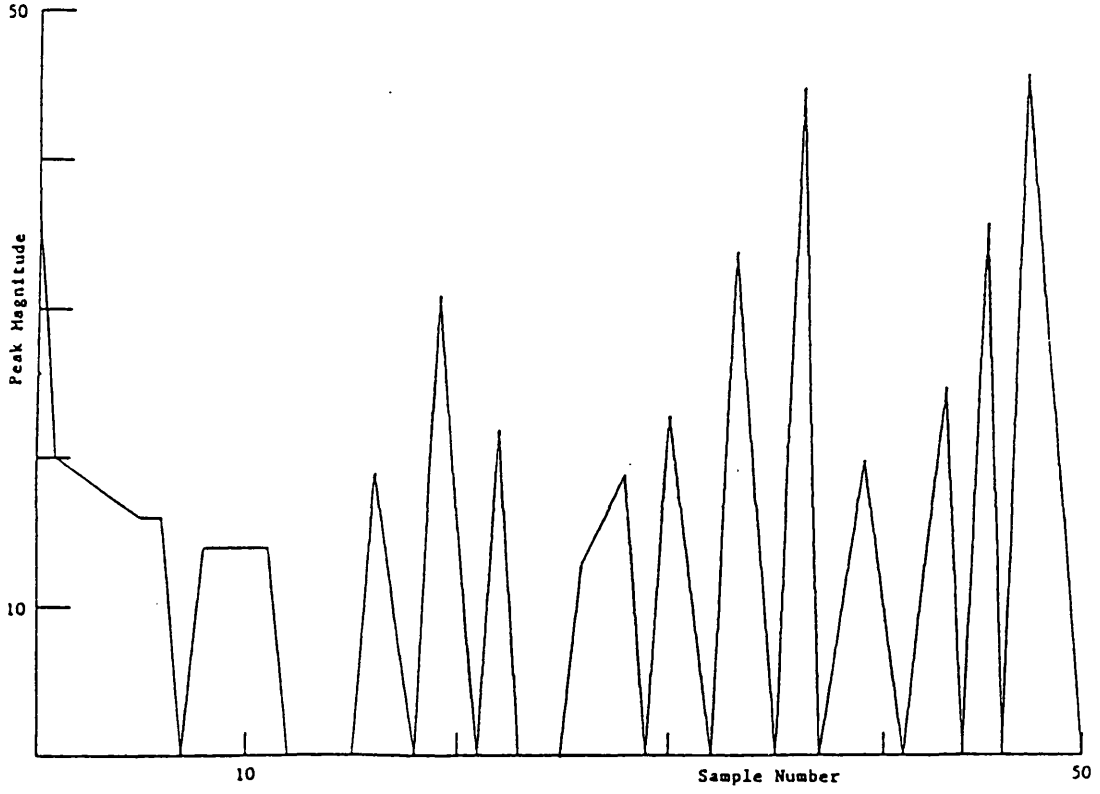


Figure 5.13 : (a) Enlarged segment of Figure 5.4(b).

(b) Equivalent segment to Figure 5.13(b) after processing by EXTR2.



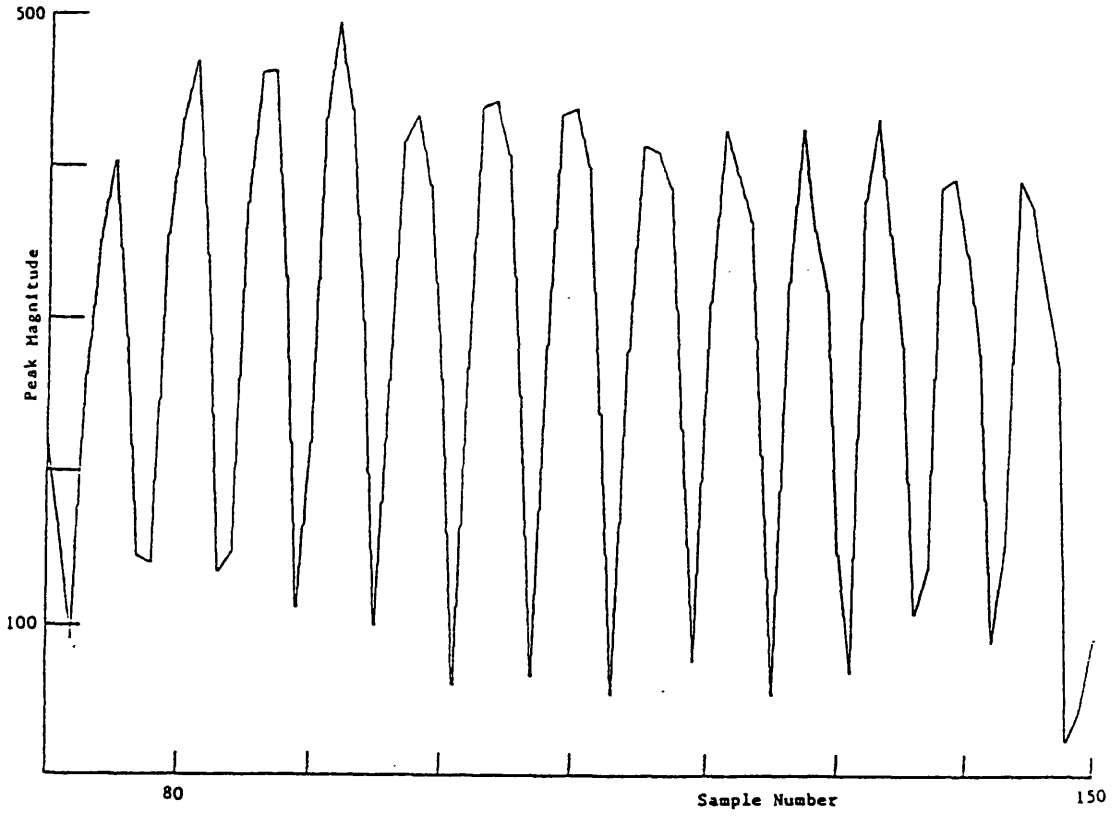
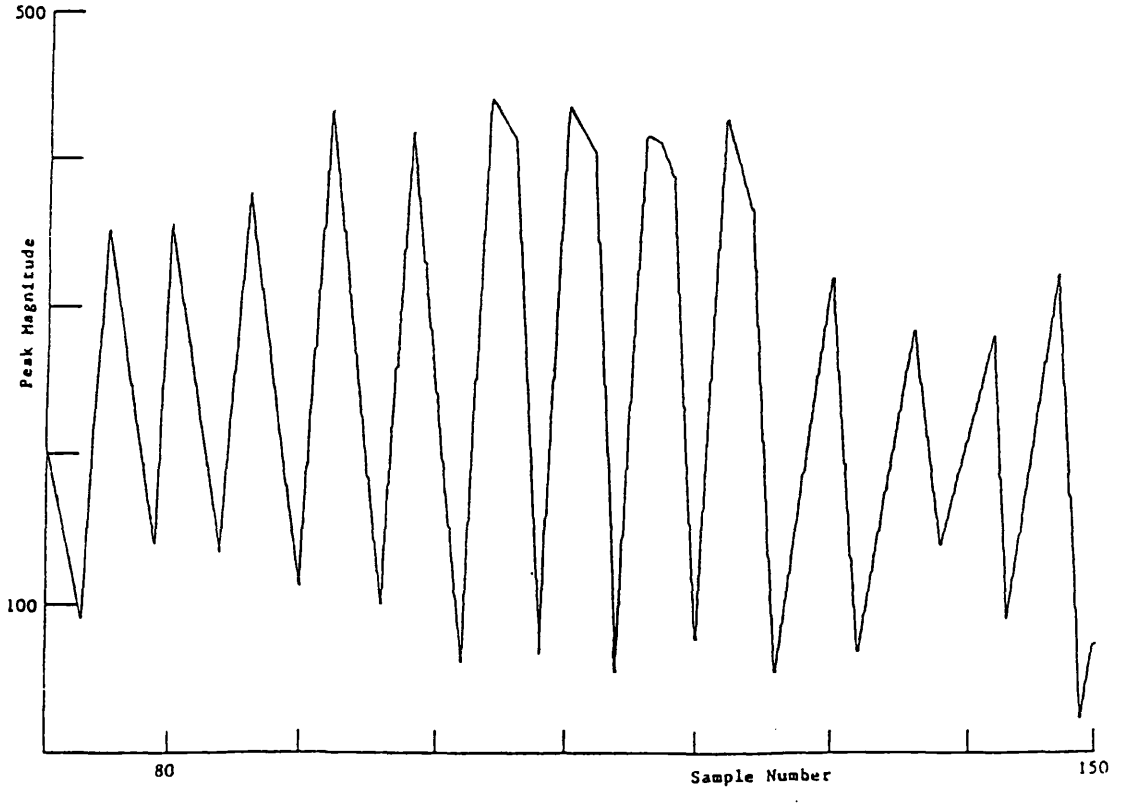


Figure 5.14 : (a) Enlarged segment of Figure 5.5(b).

(b) Equivalent segment to Figure 5.14(b) after processing by EXTR2.



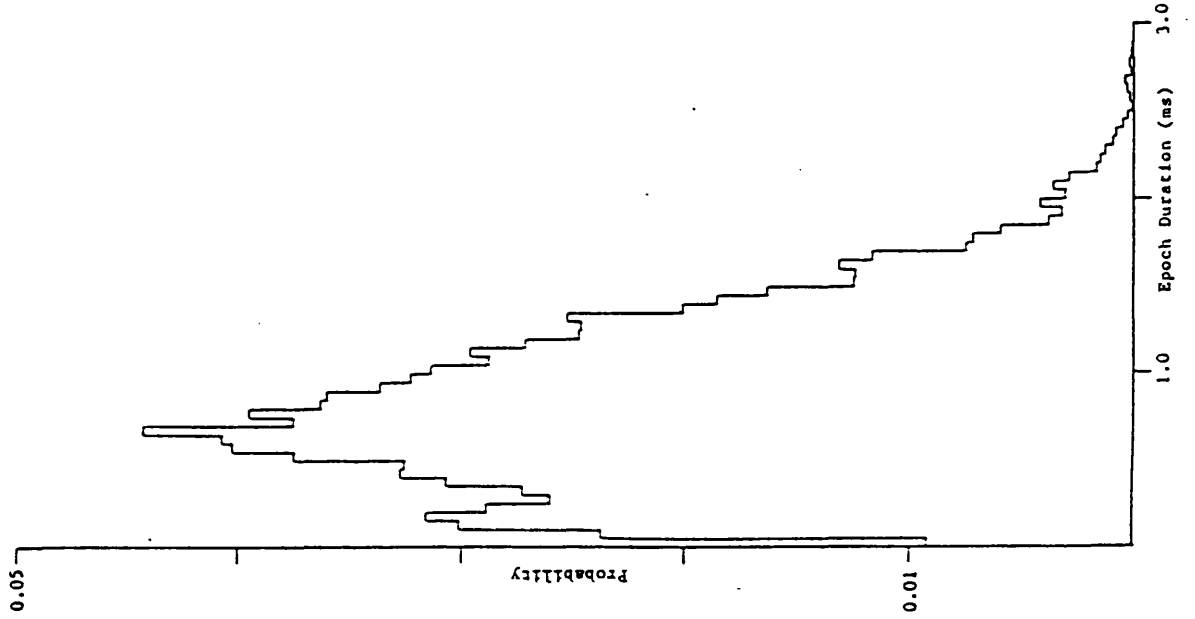


Figure 5.14(d) : Epoch duration probability distribution after extremal coding.

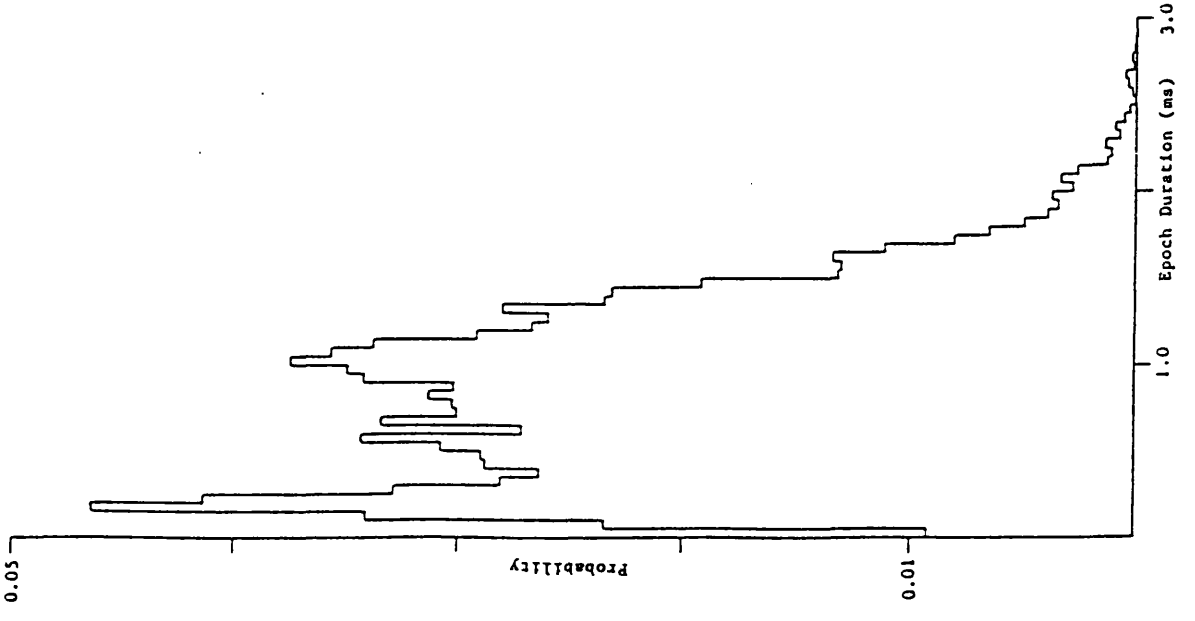


Figure 5.14(c) : Epoch duration probability distribution before extremal coding.

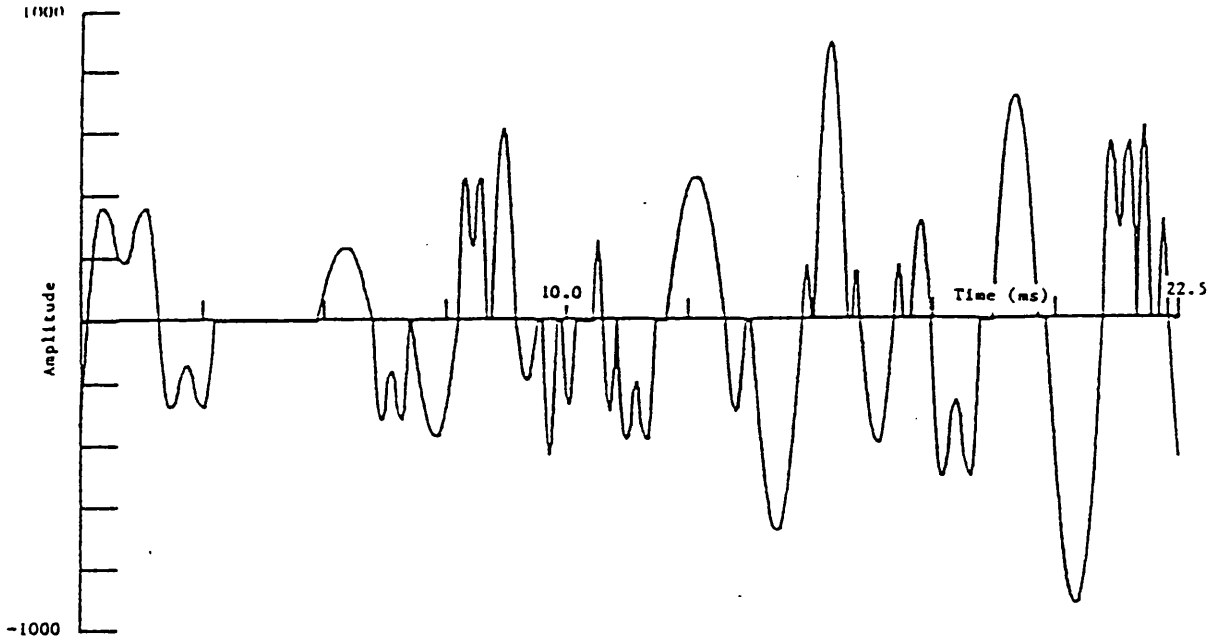
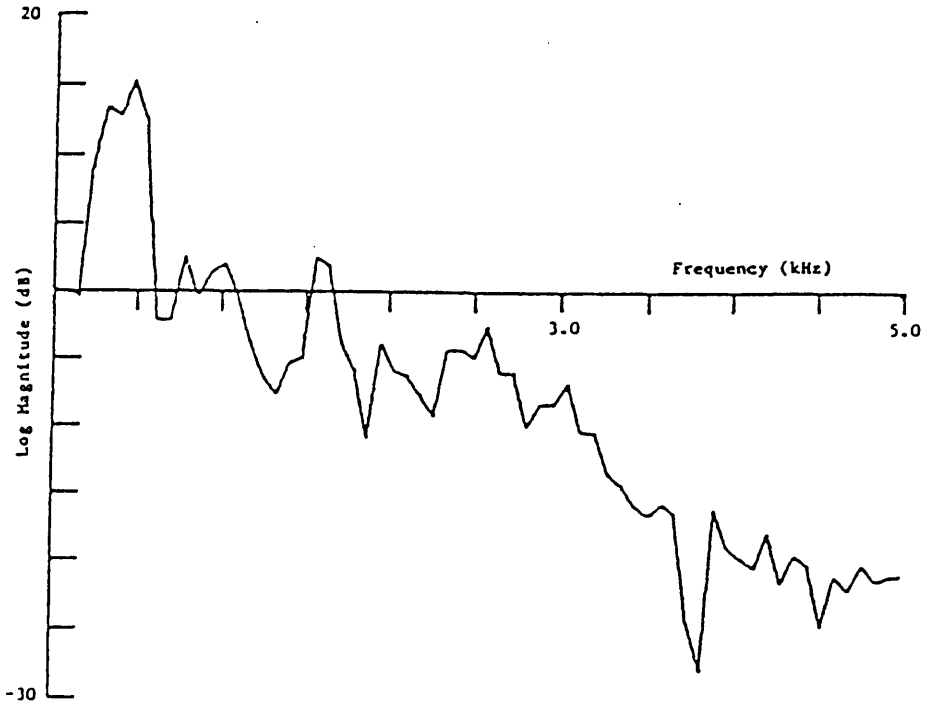


Figure 5.15 : (a) Segment of speech waveform synthesised utilising the input epoch durations of Figure 5.4(a) and the peak magnitudes of Figure 5.4(b).

(b) Corresponding power spectral density plot.



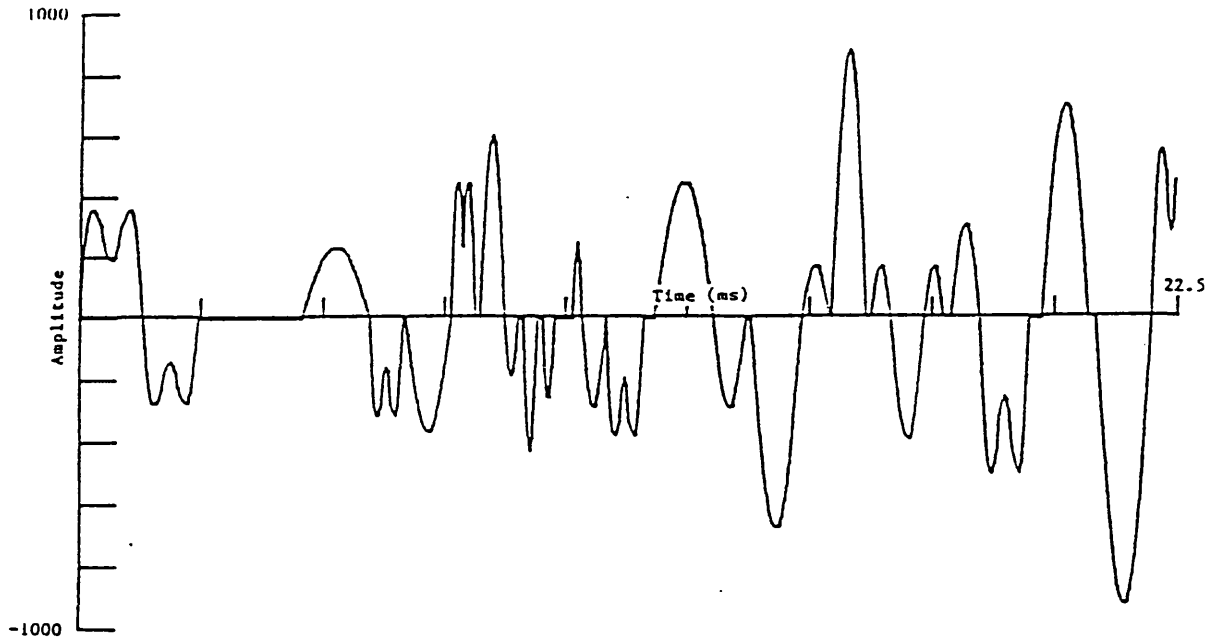
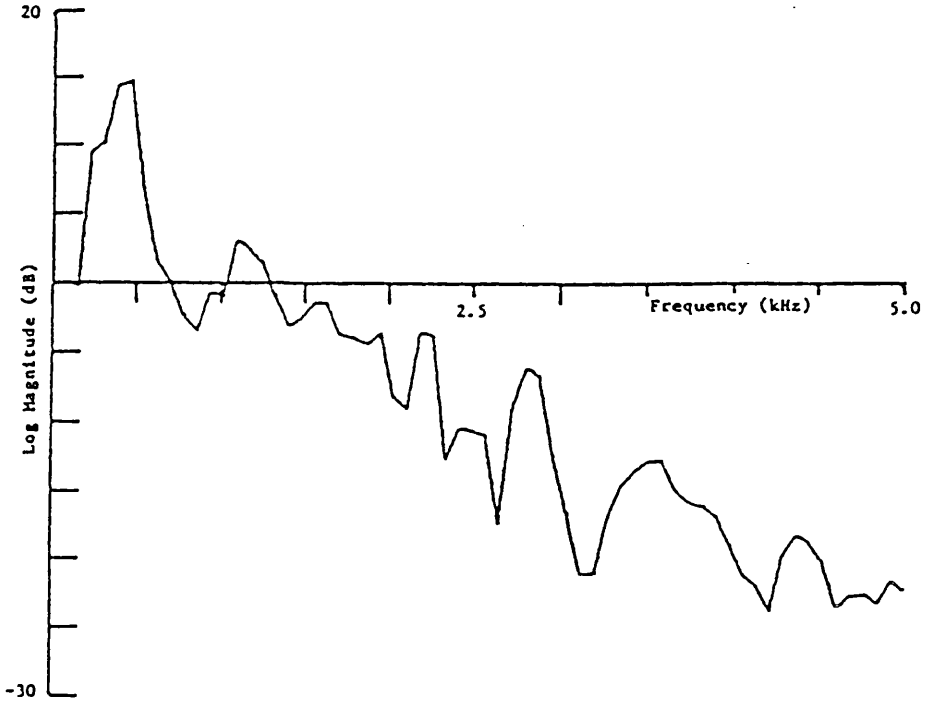


Figure 5.16 : (a) Segment of speech waveform synthesised utilising the epoch durations of Figure 5.9(b) and the peak magnitudes of Figure 5.4(b).

(b) Corresponding power spectral density plot.



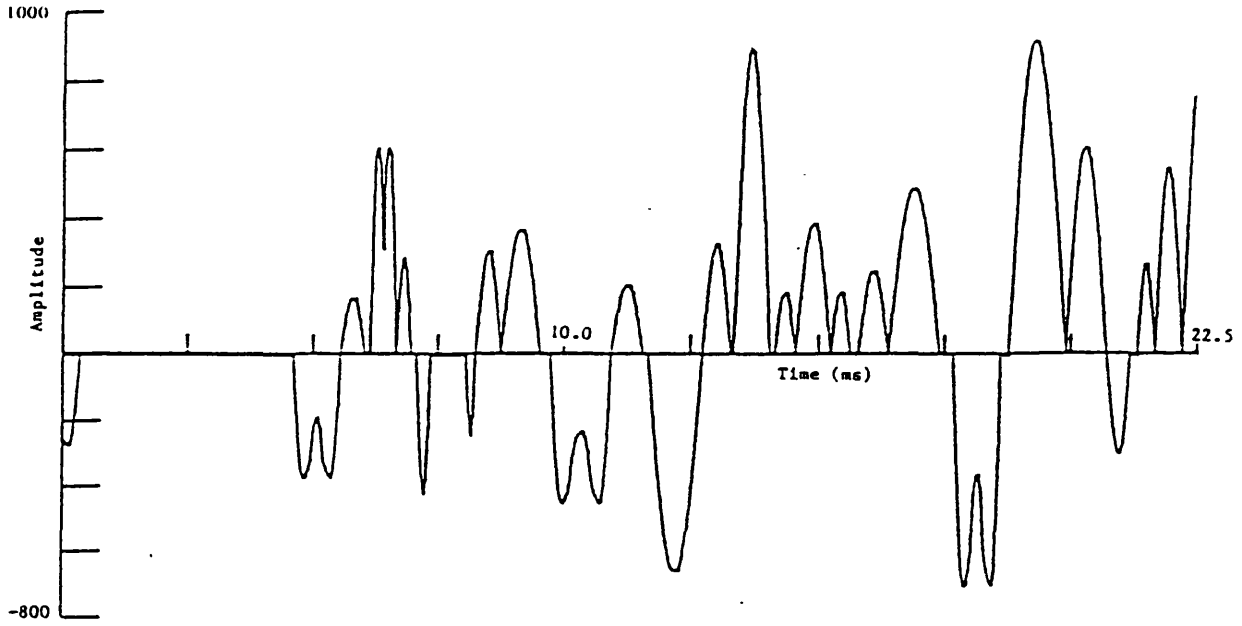
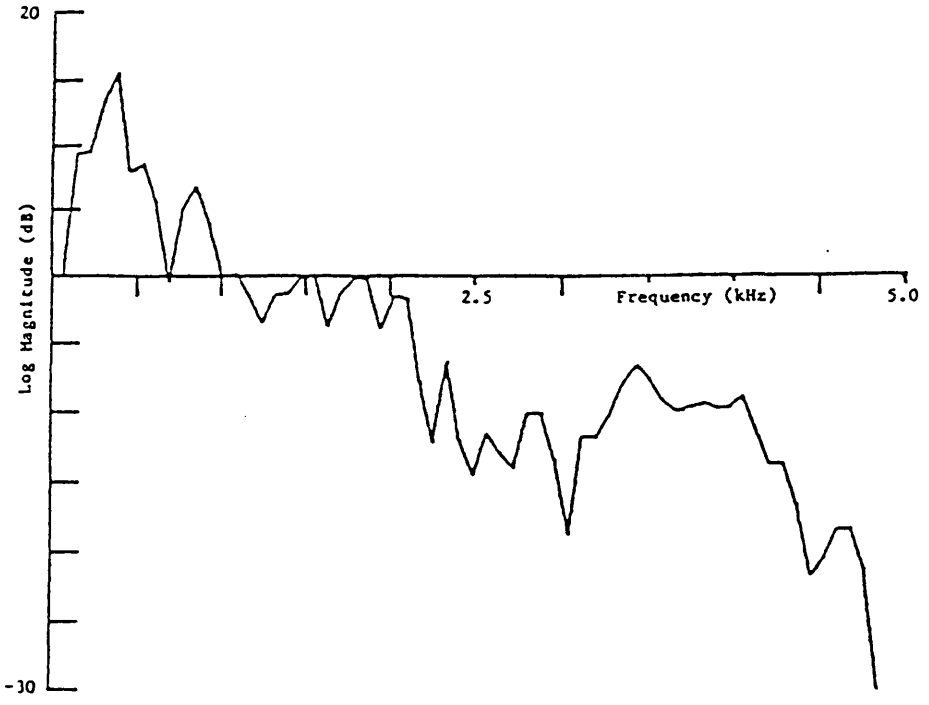


Figure 5.17 : (a) Segment of speech waveform synthesised utilising the epoch durations of Figure 5.9(b) and the peak magnitudes of Figure 5.13(b).

(b) Corresponding power spectral density plot.



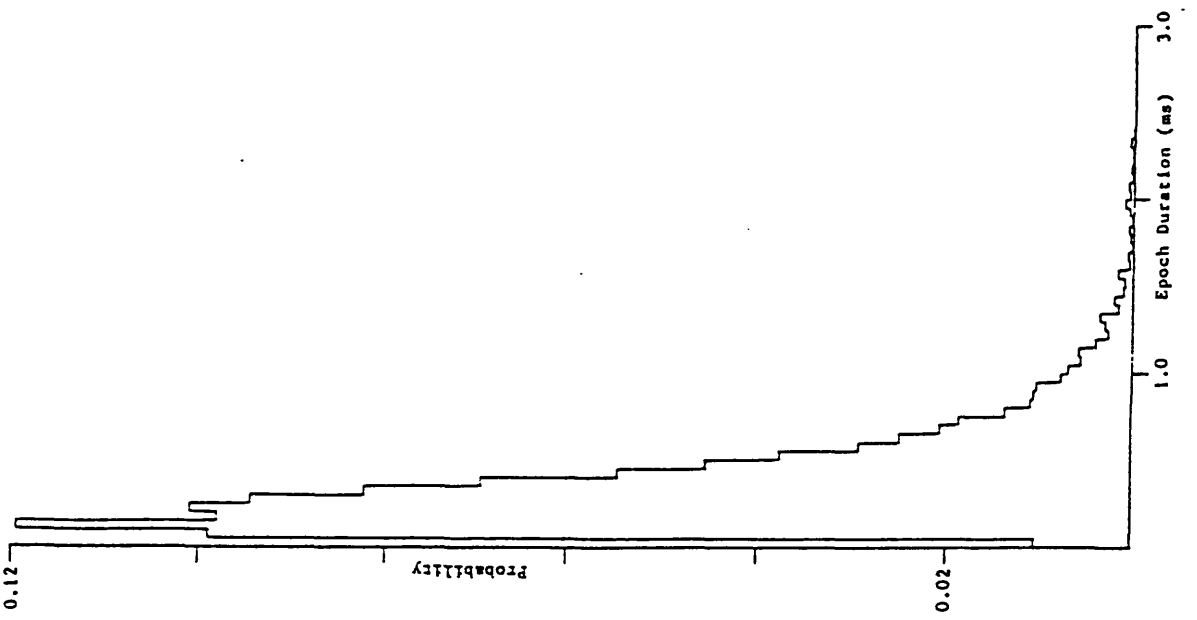


Figure 5.18(b) : Absolute value of the interpolation values probability distribution.

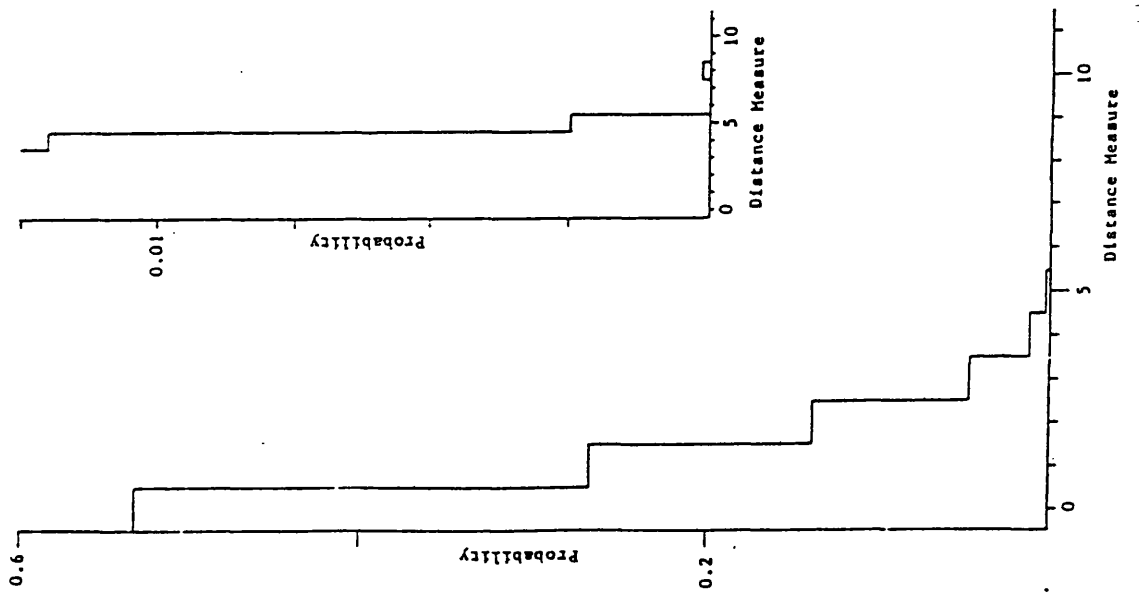


Figure 5.18(a) : Distance Measure Probability Distribution.

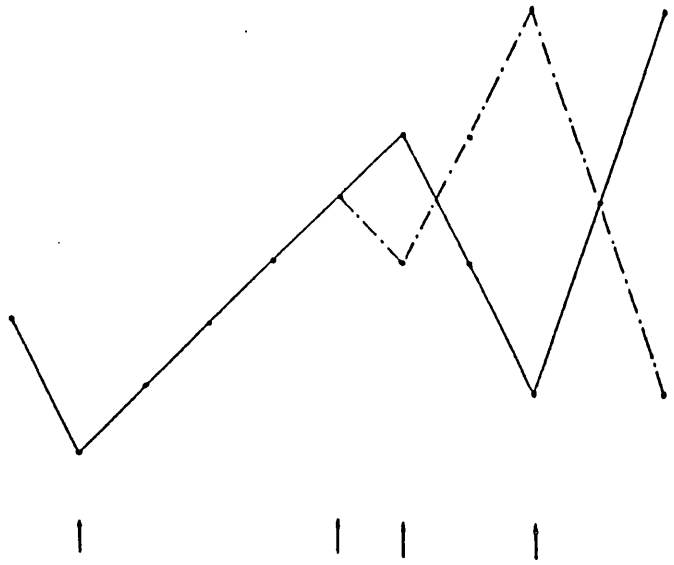


Figure 5.19(a) : Example of Extremal Coding employing a restricted distance measure of 3.

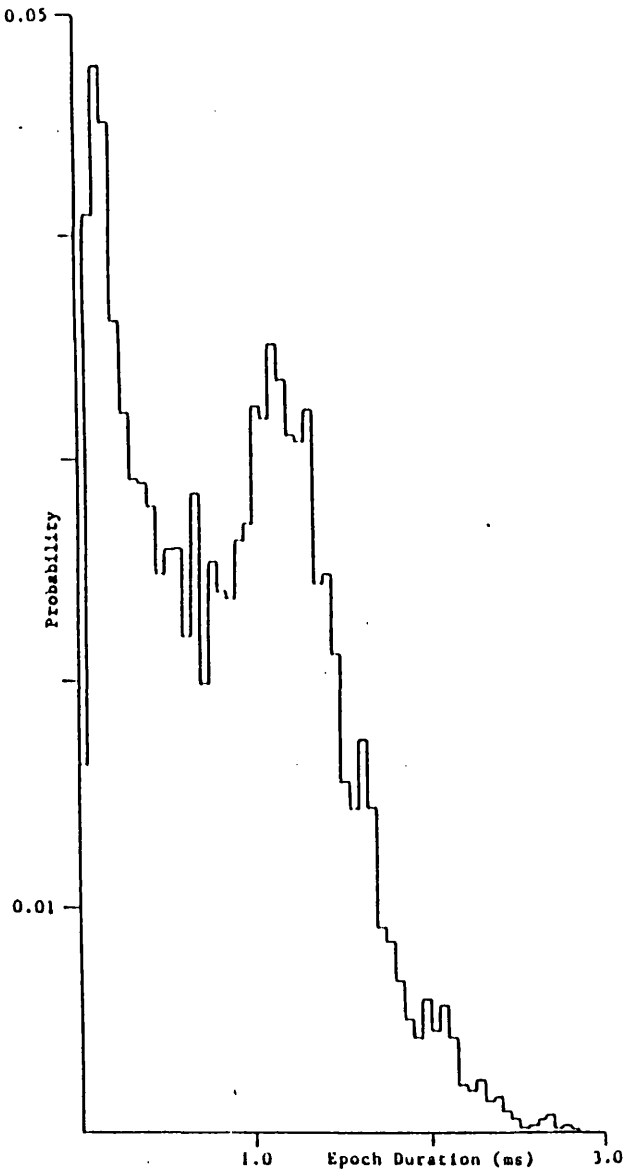


Figure 5.19(b) : Probability Distribution of epoch durations occurring as extrema.

Chapter 6

Orthogonal Transformations

6.1 Introduction

Orthogonal transformation techniques for digital speech processing have been suggested and researched to various degrees by a number of investigators [81,82,84-87]. The general approach is to choose an arbitrary sample block, N, which offers effective spectral resolution as well as computational efficiency. The performance of discrete Karhunen-Loeve (K-L), Fourier and Walsh orthogonal transformations applied to bit rate reduction for the transmission of speech signals have been measured and reported by independent researchers [84-86,88]. Their results tended to indicate that the K-L transformations offered greater efficiency in terms of data compression, while the remainder may be rated in the order they appear above.

For the K-L transformation [85], the orthogonal functions used were a special set determined from the autocorrelation functions computed over an ensemble of talkers and utterances typical of those to be processed. These have been reproduced along with the equivalent Fourier and Walsh discrete orthogonal functions in figure 6.1, for N = 16.

The K-L and Fourier transformations favour smooth-varying signals whereas the Walsh transformation favours rectangular, or more precisely, discontinuous signals. It was therefore not quite so surprising that the K-L and Fourier transformations achieved superior data reductions to that of the Walsh transformations.

The fact that Walsh transformations favour discontinuous signals gave rise to the initial impetus for these investigations. As des-

cribed earlier (chapter 4), sequences of epoch durations exhibit periods of both erratic and periodic behaviour. During erratic sequences, the differences between successive epoch durations varies considerably and this is manifested as pronounced discontinuities within the sequential plots. The segments of periodic sequences were formed by the repetition of a sequence of 6 or 8 epoch durations which were produced by the coding of voiced sounds within the input speech. The "boundary" between one sequence and the next repeat exists at the transition from an epoch duration of typically 0.2 to 0.4ms to that of 1.0 to 1.2ms long. At these points prominent discontinuities exist. To "adequately" represent sequences with such discontinuities, the K-L and Fourier transformations would require a greater number of transform coefficients in comparison to that of the Walsh transformation.

6.2 Terminology

During initial literature searches an inconsistent vocabulary pertaining to Walsh functions and transformations was found to exist. An unsuccessful attempt was made by Ahmed et al [89] to introduce a more compact and standard notation. It is therefore the purpose of appendix A5 to introduce and define the three general classifications of the continuous and discrete Walsh functions.

Using the three classes of Hadamard matrices of appendix A5, one can define the corresponding transformations as follows:

$$Y(N) = H_w(N).X(N) \qquad (6.1)$$

$$Y(N) = H_h(N).X(N) \qquad (6.2)$$

$$Y(N) = H_p(N).X(N) \qquad (6.3)$$

where X(N) is a column input vector

and Y(N) is a column transform vector

In the literature, the expression (6.3) has been referred to as the dyadic-ordered and Paley-ordered Walsh transform, while (6.1) and (6.2) have been referred to as Walsh [91], Hadamard [83,85], Walsh-Hadamard [86], Walsh-Fourier [90] and BIFORE transform [92]. To eliminate ambiguity when referring to the three possible transforms Table 6.1 was produced. This table presents the full and short-form names of each transform, the abbreviations used and the name given to the elements of the column transform vector, Y(N).

6.3 Data Reduction with Hadamard Transforms, $(WT)_h$.

A Fortran program was developed which employed the Fast Hadamard Transform (FHT) subroutine given in appendix A5. The program required the user to specify the following information:

(a) Input and Output files

(b) Number of points (samples) per transform

(c) Forward or Inverse transform required.

Using the Hadamard transform, three data reduction techniques were implemented and investigated. These techniques were incorporated into the main program as subroutines. When the user specified that the inverse transform was required a menu of coefficient processing techniques were presented to the user. The menu also included a "no action" option. The data reduction techniques were based

upon the objective of achieving a direct 2:1 data reduction, as well as spectral considerations (in the sequency sense), of the input sequence. The techniques employed are outlined below:

6.3.1 Dominant Coefficient Retention - retain N/4 + 1 coefficients.

For an N point transform only n dominant coefficients (in terms of the largest absolute value in the transform domain) were retained [86,88]. In so doing this the majority of the selected components were in the high energy region of the spectrum. This then preserved the dominant characteristics of the input sequence. Since the sequency of the dominant coefficients depends on the characteristics of the input sequence, information was also transmitted to specify the sequency positions of the coefficients retained.

6.3.2 Sequency-based Vector Filtering

Two different sequency-limited filters were implemented. The filters utilised were:

(a) Low-pass Sequency Filtering - Elimination of alternate Hadamard coefficients.

The elimination of alternate coefficients automatically reduced the data requirements by a factor of 2. It also eliminated the N/2 highest sequency components thereby low-pass filtering the input sequence.

(b) Multiple Band-pass Sequency Filtering - Elimination of the last N/2 Hadamard coefficients.

> Once again an immediate data reduction by a factor of 2 was achieved. However, in terms of sequency N/4 "stop-band", each of two sequency components, were imposed. The filter retained the highest and lowest sequency components plus mid-band components in groups of two.

Figure 6.2 depicts each of these filters for (a) the Hadamard-ordered Walsh Transform and (b) the Sequency-ordered Walsh Transform. To implement these filters, the column vector was "multiplied" by an (N×N) diagonal filter matrix before re-transformation took place. The diagonal elements of the filter matrix were limited in value to either 0 or 1.

The values of the input sequence were limited to the range of {1,64}. The discarding of coefficients before inverse transformation may have resulted in samples having values outside the permitted range of the input data. Contingencies for this eventuality were therefore incorporated into the Fortran program.

The first row of all three classes of the transform consisted of only +1's. Since the input data was limited to the range {1,64}, following a forward transformation, the first coefficient, y_0, represents the summation of the N samples of the input vector. Thus, in all situations, the first coefficient was of the greatest absolute value and represented the total duration of the N epochs of the

input vector. It was, therefore, ensured that this coefficient was not distorted by any of the data reduction techniques.

The value of y_o was recorded before the inverse transformation was performed. On completion of the inverse transformation, the new data sequence was inspected for the occurrence of values outside the permitted range, {1,64}. If none were encountered then the data was stored in the output file, otherwise further processing was conducted. If a sample value was greater than 64, then its value was automatically truncated to 64. In the event of negative or zero valued samples being encountered, the summation of all the samples, x'(i), with values greater than zero was performed. The difference, DIFF, between y_o and the resultant of the summation was calculated. If the difference was zero then all samples, x'(i), with values less than one were set to one. However, if the difference was greater than zero, and L of the N samples were less than one, then DIFF was divided by L and the resultant was rounded to the nearest integer (greater than zero). The L samples were reset to this value. These operations are summarised below:

$$DIFF = y_o - \sum_{i=1}^{N} x'(i) \quad \text{for all } x'(i) \text{ greater than zero}$$

If DIFF .EQ. 0 : Then for all x'(i) .LT. 1, x'(i) = 1

If DIFF .GT. 0 : Then LDIFF = Round {DIFF/L}

However,

If LDIFF .LT. 1, Then LDIFF = 1,

and for all x'(i) .LT. 1, x'(i) = LDIFF

6.4 Bit Allocations and Reductions

The input sequence consisted of positive integer data in the range {1,64} which required six bits per sample. It was the philosophy of these investigations not to quantise the transform coefficients. Thus distortions due to the data reduction techniques were highlighted and not masked by distortions resulting from coefficient quantisation.

Since the input data were positive integer values and numerical division did not take place in the forward transformation, the range of values to which the coefficients were limited may be estimated.

6.4.1 Maximum Coefficient Value

As stated previously, the first row of the transform matrix consisted of only +1's. If the N samples of the input vector were equal to 2^m, where m is the bit accuracy, then after transformation the first coefficient, y_o, equalled:

$$y_o = N.2^m \tag{6.1}$$

Thus for constant length codewords

$$m + \log_2(N) \text{ bits} \tag{6.2}$$

are required to accurately represent y_o.

6.4.2 Minimum Coefficient Value

Apart from the first row of the transform matrix, the elements of all other rows consist of N/2 +1's and N/2 -1's. If an input vector were composed of N/2 samples equal to 2^m, with the remaining N/2 samples equal to one, and the order of the data in the input vector was such that, positionally, the N/2 samples of one coincided with the N/2 +1 elements of the nth row. After transformation the nth coefficient, y_n, equalled:

$$y_n = -\frac{N}{2}.2^m + \frac{N}{2} \qquad (6.3(a))$$

$$= -N.2^{m-1} + \frac{N}{2} \qquad (6.3(b))$$

re-arranging equation 6.3(a),

$$y_n = -\frac{N}{2}.(2^m - 1) \qquad (6.4)$$

Thus for constant length codewords

$$\log_2(2^m - 1) + \log_2(\frac{N}{2}) \quad \text{bits}$$

or

$$\log_2(2^m - 1) + \log_2(N) - 1 \quad \text{bits} \qquad (6.5)$$

were required to accurately represent the absolute value of y_n. From equation 6.3(b), equation 6.5 may be approximated by:

$$(m - 1) + \log_2(N) \quad \text{bits} \qquad (6.6)$$

Comparing equations 6.2 and 6.6, it is observed that the positive

range of the coefficients required one bit more than the negative range. To distinguish between positive and negative coefficient values, a polarity bit would be required. Therefore, employing constant length codewords

$$m + 1 + \log_2(N) \text{ bits} \tag{6.7}$$

are required to represent the full range of the transform coefficients. As specified earlier, the input samples were represented with six bit accurracy. Therefore equation 6.7 becomes:

$$7 + \log_2(N) \text{ bits} \tag{6.8}$$

With the number of bits required to represent each transform coefficients established, it is then instructive to compare the number of bits required by the data reduction techniques of section 6.3, for the representation of a block of N epoch durations, with that originally employed.

6.4.3 Dominant Coefficient Retention - Retain N/4 + 1 Coefficients.

Theoretically, for each coefficient transmitted, its sequency position must also be transmitted. However, from previous dicussions it is known that the first coefficient, y_o, was the summation of the elements of the input vector and always had the greatest absolute value. It was not therefore necessary to transmit its sequency position. Therefore N/4 + 1 coefficients and N/4 sequency positions were transmitted. Thus for a block of N "samples" the number of bits, B, required for each transform was:

$$B = (\frac{N}{4} + 1).(7 + \log_2(N)) + \frac{N}{4}.\log_2(N) \qquad (6.9)$$

$$= \frac{N}{4}.(2.\log_2(N) + 7) + \log_2(N) + 7 \qquad (6.10)$$

Substituting values for the transform block size, N, into equation 6.10 the bit requirements per transform and per epoch was calculated. This information is summarised in Table 6.2 for $N = 2^k$, where k = 2, 3, ... ,6. Inspection of Table 6.2 revealed that the optimum transform block size (from a bit reduction piont) occurred for N = 16, where a compression ratio of 1.35:1 was achieved (4.44 bits per epoch).

6.4.4 Low-pass Sequency Filter - Elmination of Alternate Hadamard Coefficients.

For an input vector of N samples, N coefficients were produced by the transformation, of which only N/2 were transmitted. The number of bits, B, required per transformation was :

$$B = \frac{N}{2}.(7 + \log_2(N)) \text{ bits} \qquad (6.11)$$

Substituting values for the transform block size, N, into equation 6.11 the number of bits required per transform, and hence per epoch, was calculated. This information is summarised in Table 6.3 for $N = 2^k$, where k = 2, 3, ... ,6. Inspection of Table 6.3 revealed that, for a transform block size of 32 "samples", data compression was not achieved and, for block sizes greater than 32, more bits were required for the representation of the transform

coefficients than the original sequence required.

It has been shown in appendix A6 that the elements of the output vector x'(i), for the above data compression technique may also be derived as follows:

Given an input sequence

$$\ldots, x_n, x_{n+1}, x_{n+2}, \ldots \qquad \text{for } n = 0,1,2, \ldots$$

$$x'(i) = \frac{(x(i) + x(i+1))}{2} \qquad \text{for } i = 1,3,5, \ldots \qquad (6.12)$$

and $x'(i+1) = x'(i)$

From equation 6.12, it was observed that this process only involved calculating the average of two adjacent input samples and replacing them with the average value. This process required a system delay of only one input sample, a single addition and division by 2 (one bit shift left). The average value, which has an accuracy of six bits, would be transmitted and thus represent two input samples. Therefore an average of 3 bits per input sample would be employed for transmission, resulting in a data reduction ratio of 2:1.

An N point Hadamard Transform requires a system delay of N input samples, $N.\log_2 N$ additions/subtractions (employing the fast transform) and, depending on the input vector size, varying degrees of data reduction ratios were achieved, all less than 1.5:1. An interesting feature of equation 6.12 is that this technique is not restricted to transform block sizes of N, which are a power of two. In fact, any even valued length of input vector may be considered.

6.4.5 Multiple Band-pass Sequency Filter - Elimination of Last N/2 Hadamard Coefficients.

For an input vector of N samples, N coefficients were produced by the transformation, of which only N/2 were transmitted. The number of bits, B, required per transformation were:

$$B = \frac{N}{2}.(7 + \log_2(N)) \text{ bits} \qquad (6.13)$$

Comparing equations 6.11 and 6.13, it is noted that they are identical and therefore the information contained in Table 6.3 also applies to this data reduction technique.

It has been shown in appendix A6 that the elements of the output vector x'(i), for the above data compression technique may also be derived as follows:

Given an input sequence

$$\ldots, x_n, x_{n+1}, x_{n+2}, \ldots \qquad \text{for } n = 0,1,2, \ldots$$

$$x'(i) = \frac{(x(i) + x(i + N/2))}{2} \quad \text{for } i = 1,2, \ldots ,N/2 \qquad (6.14)$$

and $x'(i+N/2) = x'(i)$ over a block of N samples.

The process, defined by equation 6.14, only involves calculating the average of two input samples, and replacing those samples with the average value. For an input vector of N samples, a system delay of N/2 input samples would be incurred before the above process could be performed. Some N/2 additions and divisions by 2 (one bit

left shift's) are involved and the resulting N/2 average values, which have six bit accuracy, would be transmitted to represent the N input samples. Therefore, an average of 3 bits per sample would be employed for the transmission which represents a 2:1 data reduction.

As with the process of equation 6.12, equation 6.14 is not restricted to transform block sizes of N, where N is a power of 2. Once again, any even valued length of input vector may be processed using this technique.

6.5 Results

The results presented were output from the following processes:

Transformation of (1) N = 8,16 and 32 - retaining the N/4 + 1 coefficients of the greatest absolute value.

(2) N = 16 - retaining alternate coefficients C_i, i = 0,2,4, ... , N-1

(3) N = 16 retaining the first N/2 coefficients C_i, i = 0,1,2, ... , N/2 - 1

A series of different transformation sizes were implemented for process (1). Table 6.2 shows the optimum transform size, N, to be sixteen for which only 4.44 bits per epoch were required. The next best value for N was eight, for which 4.5 bits per epoch were required followed by a third best transform size of N = 32, which resulted in 4.625 bits per epoch being utilised. These three transform sizes were therefore investigated.

It was observed that during periodic segments of epoch sequences, the periodic structure comprised, in general, six or eight epoch durations. Section 6.4, and appendix A6, demonstrated that the same result would be achieved with process (2) irrespective of the transform size, N. However, in process (3), N specified the delay over which the epoch "average and repeat" operation occurred and the number of epochs to be repeated, ie. N/2. A transform size of 8 would have involved the inclusion of a maximum of one complete periodic segment. For process (3), this would have introduced severe distortion because epochs at the beginning of the transform window would have been averaged with the epochs beyond the centre of the transform window and these would have a significantly different value. To have adopted a transform window of 32 would have involved the averaging of epoch durations separated by 16 "samples". In such a window, approximately 3 repetitions of an epoch sequence may exit. Therefore, it is possible that the transform might average the original sequence with the second repetition of the epoch sequence and also average the first repetition of the epoch sequence with the third repetition. However, the correlation between neighbouring repetitions will, in general, be greater that that of repetitions which are separated by a repetition. Therefore, distortions will be introduced by a transform window of 32. Also, if a transition from periodic to erratic epoch durations, or visa versa, occurs within the transform window then the periodic segment would be averaged with the erratic segment resulting in distortion of the periodic sequence. It was therefore decided to employ a transform window of N = 16. Thus, in general, neighbouring periodic segments would be

averaged and the distortions which arise during the transition between the different type of epoch duration sequences would be reduced. Since the distortions introduced by process (2) are independent of transform size, N, a transform of N = 16 was also employed.

Two segments of epoch duration sequence with differing characteristics were chosen for detailed comparison. One was of an erratic nature, figure 6.3(a), while the other exhibited a strong periodic element, figure 6.3(b). To perform the comparisons the epoch duration sequences of the output files which corresponded to figures 6.3(a) and (b), were isolated and plotted. Segments of speech were synthesised using the processed CSFs which corresponded to the unprocessed CSF of figure 6.3(b). Employing a Digital Fast Fourier Transformation software package, the short-term power spectral density was obtained and plotted for comparison purposes.

Section 6.5.1 discusses the output epoch duration sequences, section 6.5.2 presents an informal subjective appraisal of the synthesised utterances and, finally, section 5.3 inspects the power spectral density plots. Table 6.4 provides a cross reference between the processes utilised and the diagrams presented.

6.5.1 Epoch Duration Sequence Comparisons

Figures 6.3(a) and 6.3(b) present the epoch duration sequencies input to the transform algorithms.

The epoch duration sequences output from process (1) (Dominant Coefficient Retention) with N = 8 are presented in figure 6.4(a) and

(b). Comparisons of figures 6.3(a) and 6.4(a) indicated that the envelope of the original sequence was retained, albeit slightly attenuated. As expected, varying degrees of distortion of the epoch durations were experienced but the erratic nature of the input sequence appeared to have been preserved. Sample 17 to 40 of figure 6.4(a) exhibited a "step-like" characteristic. A more detailed examination revealed that this was a similar effect to that which occurs due to low-pass sequency filtering. With a transform size, N, of eight this effect occurred on three consecutive transforms. Examination of the data revealed that the dominant coefficients of the transform were the same as those which would have been utilised by a low-pass sequency filtering transform. Therefore, these occurrences were a function of the data and the similarity was purely coincidental. The epoch duration sequence of figure 6.4(b) retained the general envelope of the original sequence (figure 6.3(b)). However, no obvious periodic characteristics could be perceived. Varying degrees of sample distortion occurred and these eliminated the periodicity which was so predominant within the original sequence.

The epoch duration sequences of figures 6.5(a) and (b) were produced utilising process (1) (Dominant Coefficient Retention) with N = 16. The sequence of figure 6.5(a) retained the envelope of the original sequence plus the erratic characteristics. However, an interesting feature emerged over the samples of 20 to 25, 33 to 36 and 37 to 48. This feature either "Assymmetric" or "Symmetric Mirror Imaging". Assymmetric Mirror Imaging occurred when samples $x_{n-1} = x_{n+1}$, $x_{n-2} = x_{n+2}$, ... ie. the samples were centred about the sample at x_n. However, Symmetric Mirror Imaging occurred when $x_n = x_{n-1}$,

$x_{n+1} = x_{n-2}$, ... ie. the samples were centred about a point halfway between samples x_n and x_{n-1}. All three examples cited above were of symmetric mirror imaging ie. samples 33 and 36, 34 and 35 were equal in value. This feature also occurred in figure 6.5(b) over the samples 9 to 16 an 18 to 31. Another feature, termed "delayed sample repetition" can be seen in figure 6.5(b). Samples 32 to 36 were repeated at samples 38 to 42. This feature was predicted to occur due to the multiple band-pass sequency filtering (process (3)) of the epoch duration sequence. However, this occurrence was not due to process (2) because the repeated sequence was separated from the original by a "spurious" sample, No. 37. Apart from these "patterns" a periodic characteristic was not evident within figure 6.5(b).

For N = 32 in process (1), the epoch duration sequences of figures 6.6(a) and (b) were produced. The sequence of figure 6.6(a) retained the envelope of the input waveform and had an erratic characteristic. Asymmetric mirror imaging centred on sample number 51, over the range of samples 48 to 55 occurred. Otherwise, apart from the expected sample distortions, no other features existed. The sequence of figure 6.6(b) preserved the envelope of the original sequence. Symmetric mirror imaging occurred between samples 36 to 61. It was demonstrated in appendix A6, section 2(ii) that symmetric mirror imaging would be produced when alternate coefficients of a Sequency-Ordered Walsh Transform were discarded ie. sequency of 1, 3, 5 and 7 for N = 8. However, in the Hadamard domain this does not manifest as a pattern and therefore cannot be attributed to any particular feature of the Dominant Coefficient Retention technique.

For N = 16, the Low-pass Sequency Filtering (process (2)) produced the epoch duration sequencies of figures 6.7(a) and (b). As shown in appendix A6, these sequences are identical to that achieved by calculating the average of two ajacent epoch durations and replacing the original values with the averaged values. Thus the sequences exhibit a square/step-like characteristic. Much of the erratic form of figure 6.3(a) was eliminated (figure 6.7(a)). However, the envelope of the original sequence was retained. Figure 6.7(b) shows that a periodic characteristic of the original sequence (figure 6.3(b)) was preserved but the shorter epoch durations (less than 0.5ms) were severely distorted.

Multiple Band-pass Sequency Filtering (process (3)) with N = 16 produced the epoch duration sequences of figures 6.8(a) and (b). Without prior knowledge of appendix A6 it was not immediately obvious that samples 9 to 16 were identical to those of 1 to 8 for each transform window. The sequence of figure 6.8(a) appear to have retained its erratic form but its envelope is not so well defined. The periodic characteristic of figure 6.3(b) was almost non-existant in figure 6.8(b). The majority of epoch durations had values in the range of 0.5 to 0.75ms and any periodic structure perceived may be attributed to the average and repeat process.

6.5.2 Informal Subjective Appraisal

The speech synthesised employing the unprocessed coded speech file (CSF) (the data input to the transform algorithms) was of high quality and intelligibility. The description of distortions in the

speech synthesised utilising processed CSFs are therefore relative to that produced employing the unprocessed CSF.

The CSFs output from the transform algorithms were utilised as the input files for the speech synthesis algorithm described in chapter 2. The synthesised speech output was bandlimited (300 to 3400Hz).

The synthesised speech produced, using the CSF output from process (1) with N = 8, was intelligible and time warping was very evident. Due to the distortions of the epoch durations inflicted by the transformations, the time duration of the synthesised utterance was less than that of the original. This caused the speech to sound "speeded up". Some granular noise was perceived in the background but otherwise the speech was of good quality.

The synthesised speech created, utilising the CSF from process (1) with N = 16, was, marginally, of inferior quality to that produced with N = 8. Once again, granular noise was evident and time warping of the speech had occurred. The male utterance of the word "Balam" was muffled. This disturbance was not significant enough to cause loss of word intelligibility but sufficient enough for it to be highlighted by the remainder of the utterance and therefore give an impression of reduced quality to the complete utterance.

The synthesised speech generated from the CSF output of process (1) with N = 32 was found to contain some low frequency disturbances. This caused masking of the true quality of the synthesised speech. The low frequency cut-off of the band-pass filter was increased to

400Hz before the disturbances were eliminated. The resulting speech was however severely time warped to such an extent that truncation of the last word of the complete utterance occurred. These effects resulted in a loss of naturalness within the speech, which in turn gave an impression of reduced quality in comparison with that produced for N = 8 and 16 in process (1). Word and sentence intelligibity were retained.

The speech synthesised using the CSF output from process (2) was of high quality and intelligibility. Some granular background noise at a very low level was detected, but its level was such that it did not distract the listeners attention from the utterance.

The utterance synthesised employing the CSF output of process (3) had a gargled/muffled quality, this being more pronounced in the male utterance of "Balam". The female utterance of the word "yes" was less 'crisp' than the original and had a "flat and drawled" quality. The /s/ of "yes" was obscurred by noise. Time distortion was not evident but any such occurrences may have been masked by the poor quality and reduced intelligibilty.

6.5.3 Power Spectral Density Measurements

When the processed epoch duration sequences were input to the speech synthesis algorithm the distortions in the sequences (due to the transformations) manifested as time distortions of the waveform. As described in the previous section, the resulting distortion, termed "time warping", was predominantly of the variety which caused

the complete utterance to occur over a shorter time period. Distortions of this nature caused variations in the power spectrum over bands of frequencies. However, due to the non-linear nature of the distortions introduced by the transformations implemented (Tes-domain to Time-domain) these effects cannot be catagorised nor can the frequecies affected be predicted. The time distortions were a function of the data, the bit reduction and transformation algorithms which cannot be compensated for be post filtering.

Inspection of the power spectral density plots (PSDP) revealed that the majority of the differences, in comparison with the original spectra, occurred over the mid-band frequencies (1.5 to 3.0kHz). These differences were, in general, an increase in the power spectra of the synthesised speech.

Figures 6.9(a) and (b) presented the speech synthesised from the epoch duration sequence of figure 6.3(b) and the computed power spectrum. A general comparison of power spectra revealed that the power spectrum of the speech segment synthesised using the Low-pass sequency filtered epoch durations, figure 6.13(b), had incurred the least distortions. The second spectral peak was attenuated while the midband frequencies were amplified. The distortions of the power spectra produced from the speech synthesised utilising the epoch durations output from the Dominant Coefficient Retention algorithm, appear to have increased as the transform size, N, increased. The second spectral peak was merged into the spectrum for transforms of 8 (figure 6.10(b)) and 16 samples (figure 6.11(b)). For the transform of 32 samples the second spectral peak began to separate

from the power spectrum (figure 6.12(b)). The power spectra resulting from the Multiple Band-pass Sequency Filtered epoch durations had incurred the greatest distortions with both the first and second spectral peaks experiencing frequency shifts and/or attenuation to a greater extent than any of the other power spectrums, figure 6.14(b).

6.6 Conclusions

A maximum data reduction of 1.35:1 was achieved employing Dominant Coefficient Retention with N = 16 in a Hadamard Transform. The resulting speech was intelligible but of moderate quality.

The speech synthesised from the Low-pass Sequency Filtered Hadamard coefficients was of high quality and intelligibility. A maximum data reduction of 1.33:1 was achieved for N = 4. However, analysis of this process demonstrated that an identical epoch duration sequence (to that output from the transform) is achieved if the average of pairs of adjacent epoch durations was calculated and the original epoch durations were replaced with the average value. This process yielded a 2:1 data reduction in the epoch duration sequence because only the average value need be transmitted. This process also reduced the system delay from N epoch durations to two epoch durations, where N is the size of the Hadamard transform. ie by a factor of N/2.

Transform Name	Short Form Name	Abbrev.	Equ^{n} No.	Elements of Y(N)
Walsh - Ordered Walsh Transform	Walsh Transform	$(WT)_w$	6.1	Walsh Coefficients
Hadamard-Ordered Walsh Transform	Hadamard Transform	$(WT)_h$	6.2	Hadamard Coefficients
Paley - Ordered Walsh Transform	Paley Transform	$(WT)_p$	6.3	Paley Coefficients

Table 6.1 : Summary of terms used to refer to the three classes of transform.

Transform Size,N	$Log_2(N)$	Bits Per Transform,B	Bits Per Epoch	Data Compression Ratio
4	2	20	5.00	1.2 : 1
8	3	36	4.50	1.33 : 1
16	4	71	4.44	1.35 : 1
32	5	148	4.63	1.3 : 1
64	6	317	4.95	1.21 : 1

Table 6.2 : Summary of data compression achievable utilising Dominant Coefficient Retention.

Transform Size,N	$Log_2(N)$	Bits Per Transform,B	Bits Per Epoch	Data Compression Ratio
4	2	18	4.5	1.33 : 1
8	3	40	5.0	1.2 : 1
16	4	88	5.5	1.09 : 1
32	5	192	6.0	1 : 1
64	6	416	6.5	1 : 1.08

Table 6.3 : Summary of data compression achievable utilising either Low Pass or Multiple Band Pass Sequency Filtering.

	Transform Size, N	Erratic Epoch Sequence	Periodic Epoch Sequence
Input sequence	-	6.3(a)	6.3(b)
Dominant Coeff. Retention	8	6.4(a)	6.4(b)
	16	6.5(a)	6.5(b)
	32	6.6(a)	6.6(b)
Low Pass Sequency Filter	16	6.7(a)	6.7(b)
Multiple Bandpass Sequency Filter	16	6.8(a)	6.8(b)

Table 6.4 : Cross reference of figure numbers.

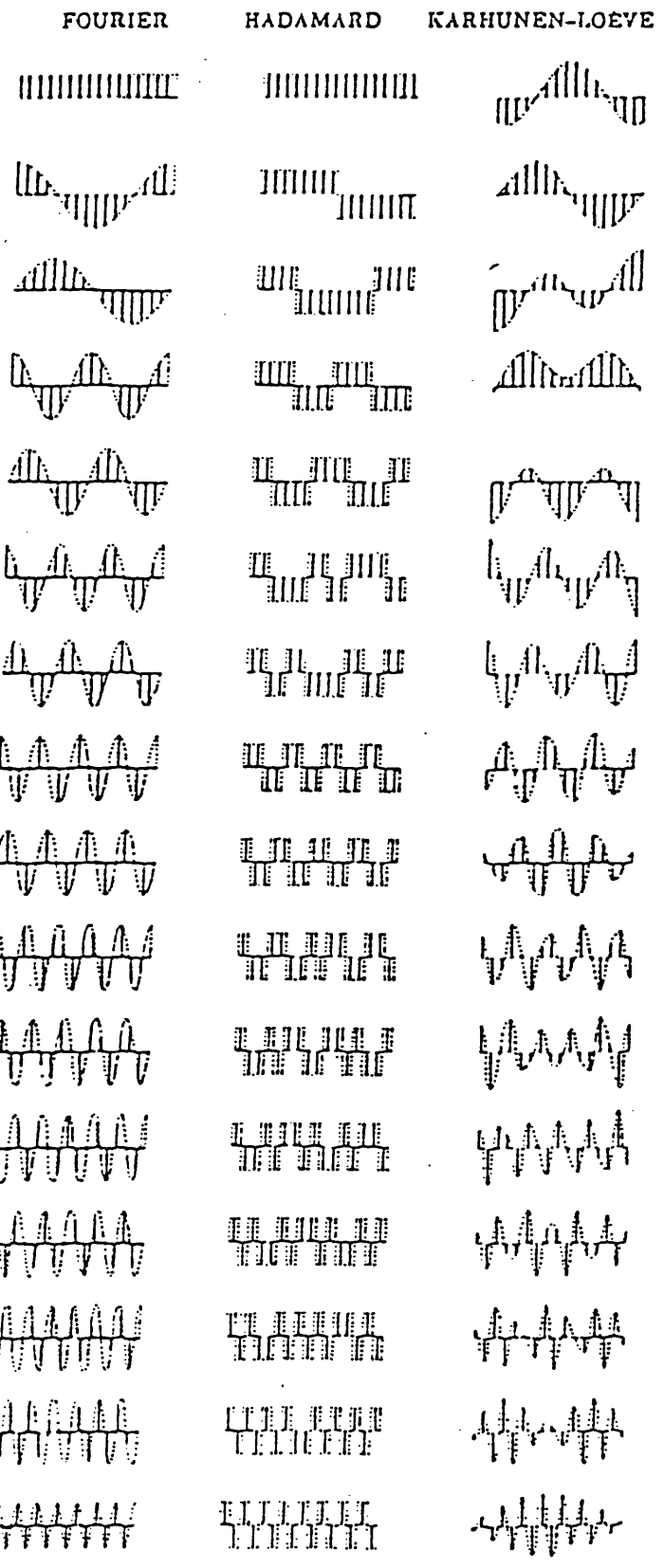


Figure 6.1 : Discrete Orthogonal Functions, N= 16 (After Campanella and Robinson [84]).

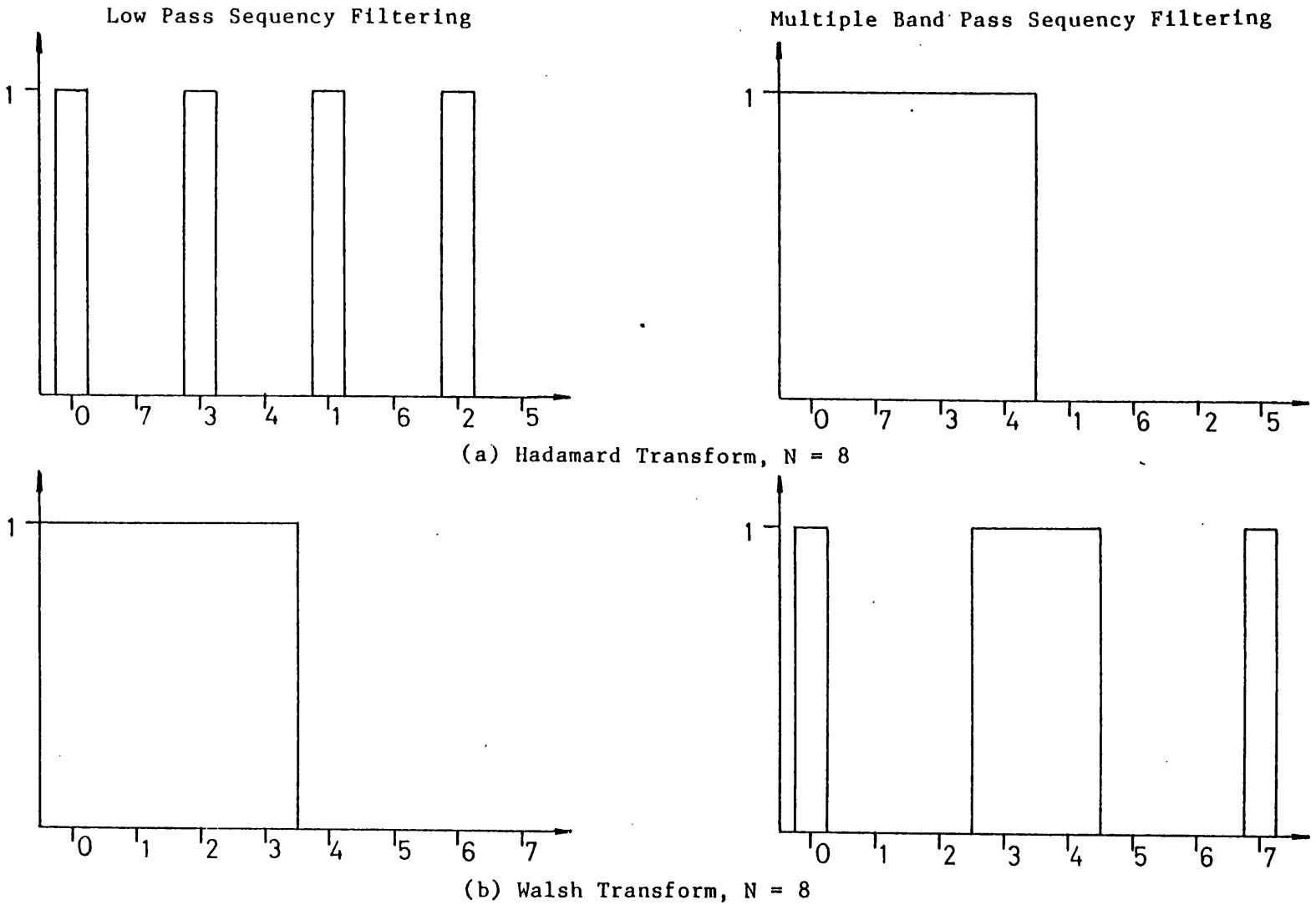


Figure 6.2 : Hadamard Transform Sequency-based Vector filtering and the equivalent Walsh Transform Sequency-based Vector filtering.

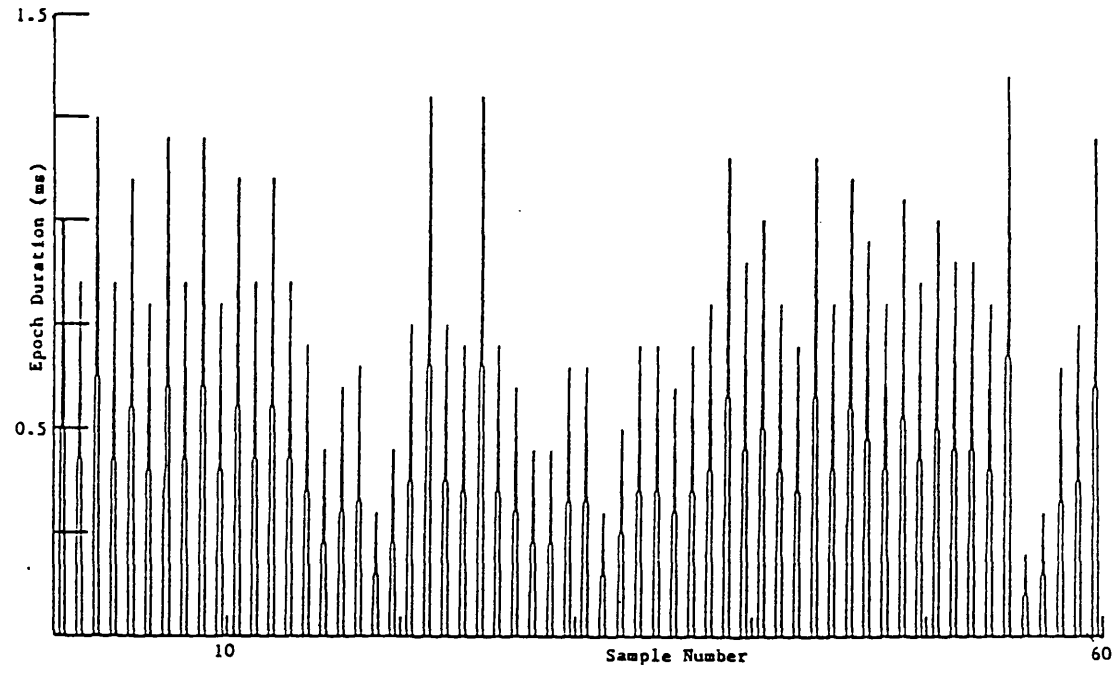
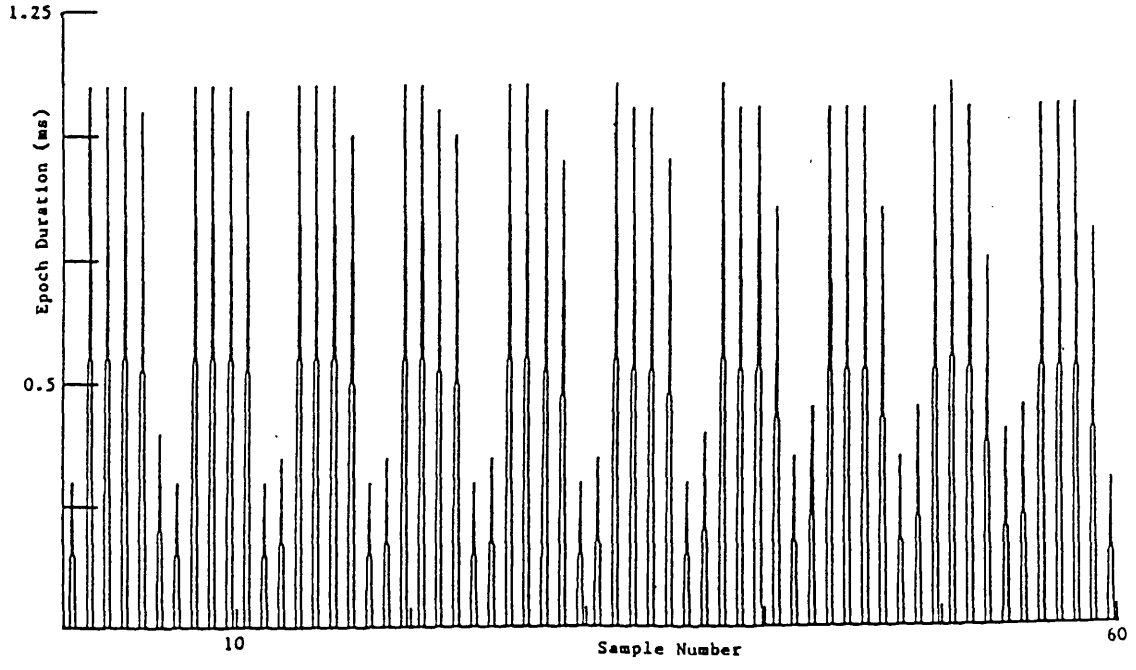


Figure 6.3(a),(b) : Discrete sequential plots of epoch durations output from Al-Doubooni's TES coder.



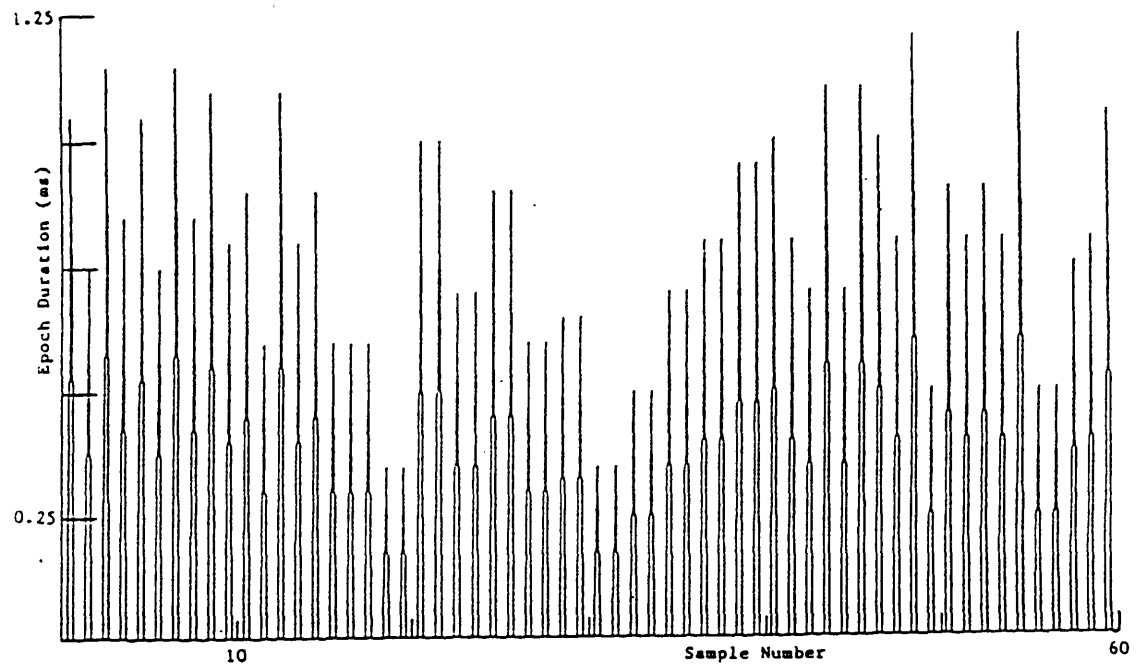


Figure 6.4 : (a) Sequence of Figure 6.3(a) after Forward and Inverse Hadamard transformation (N = 8) employing Dominant Coefficient Retention.

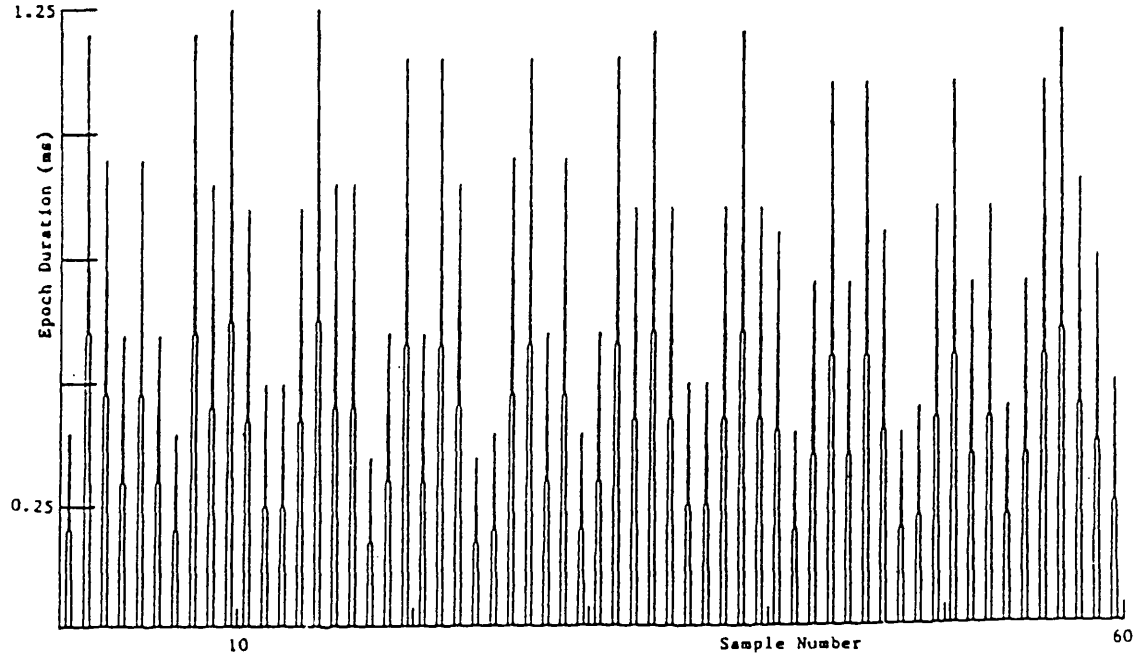


Figure 6.4 : (b) Sequence of Figure 6.3(b) after Forward and Inverse Hadamard transformation (N = 8) employing Dominant Coefficient Retention.

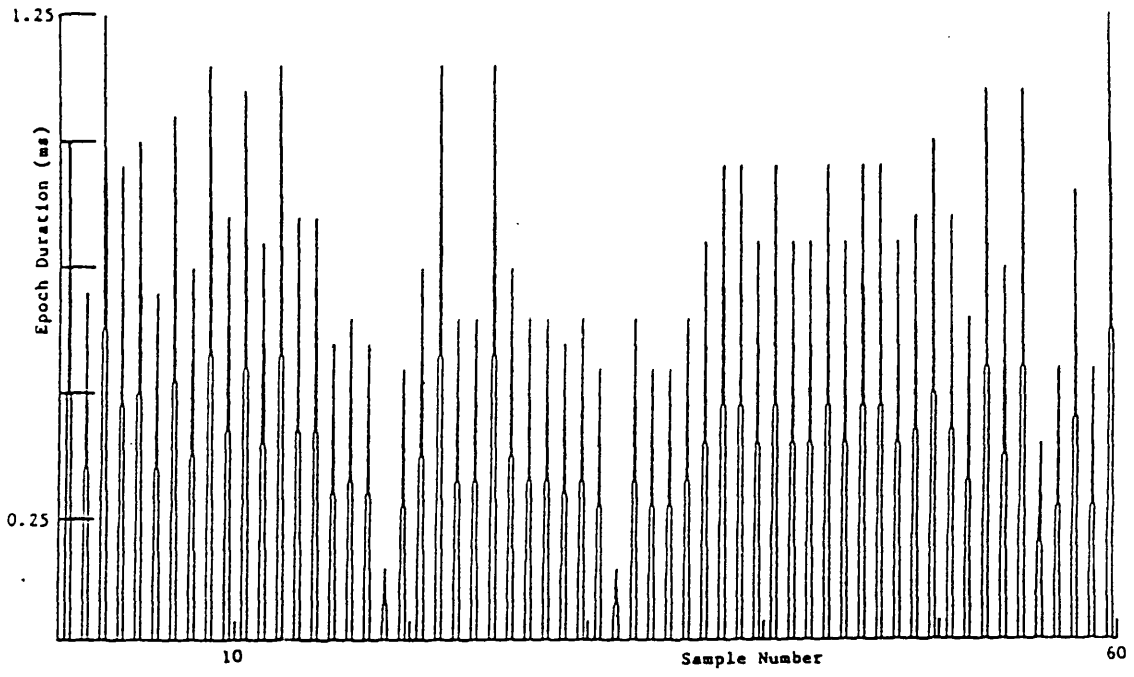


Figure 6.5 : (a) Sequence of Figure 6.3(a) after Forward and Inverse Hadamard transformation (N = 16) employing Dominant Coefficient Retention.

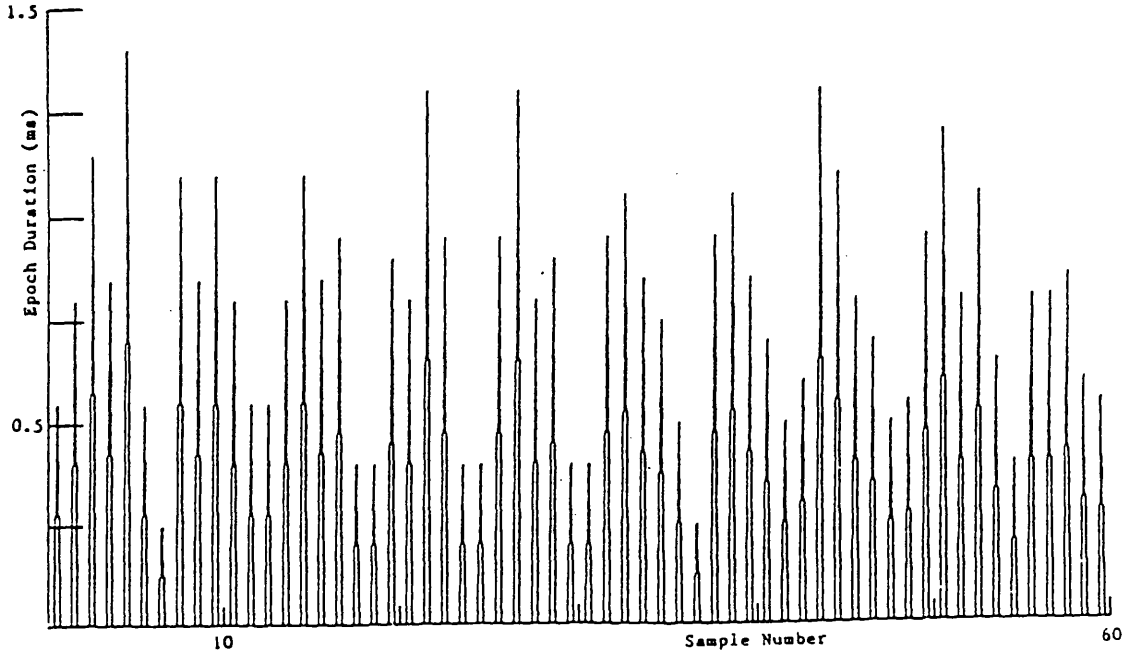


Figure 6.5 : (b) Sequence of Figure 6.3(b) after Forward and Inverse Hadamard transformation (N = 16) employing Dominant Coefficient Retention.

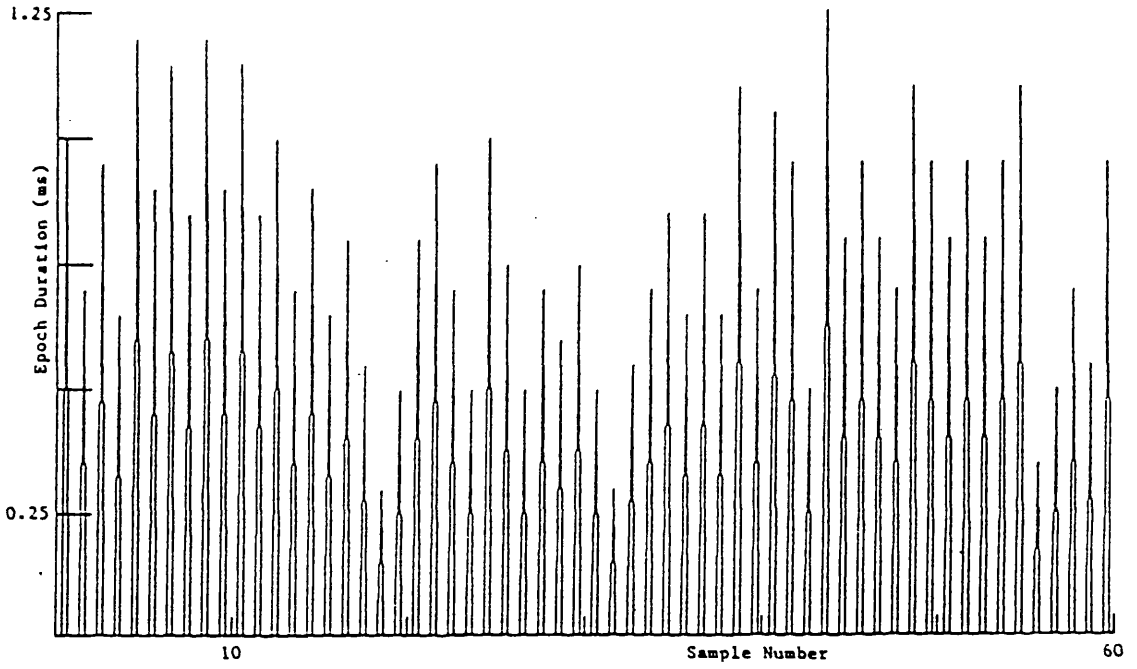


Figure 6.6 : (a) Sequence of Figure 6.3(a) after Forward and Inverse Hadamard transformation (N = 32) employing Dominant Coefficient Retention.

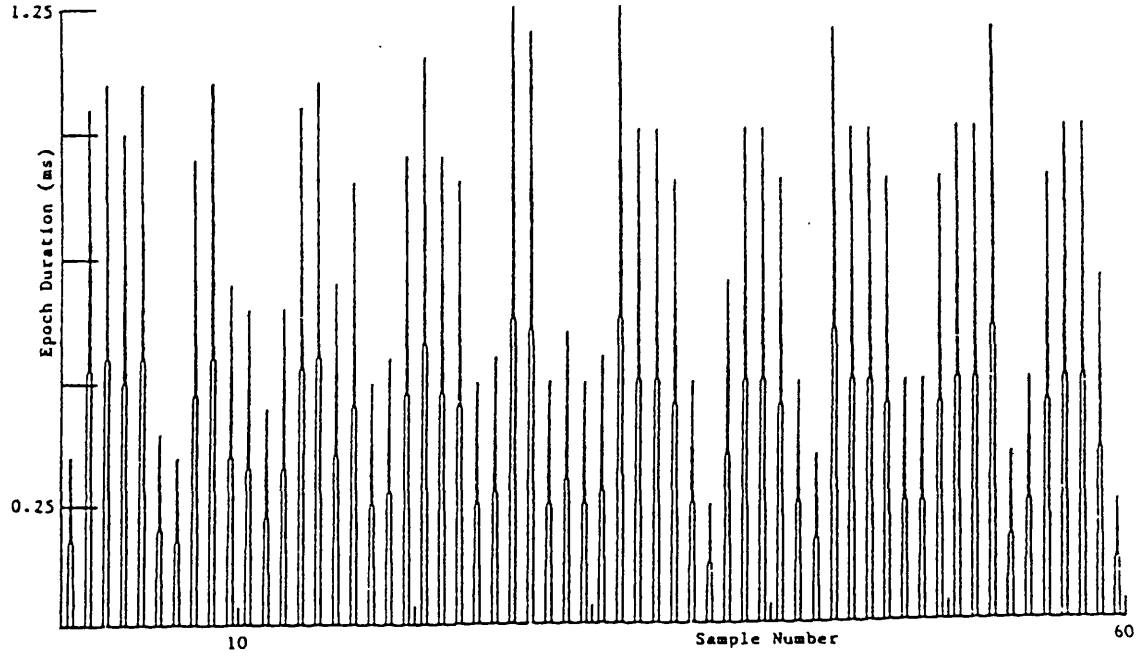


Figure 6.6 : (b) Sequence of Figure 6.3(b) after Forward and Inverse Hadamard transformation (N = 32) employing Dominant Coefficient Retention.

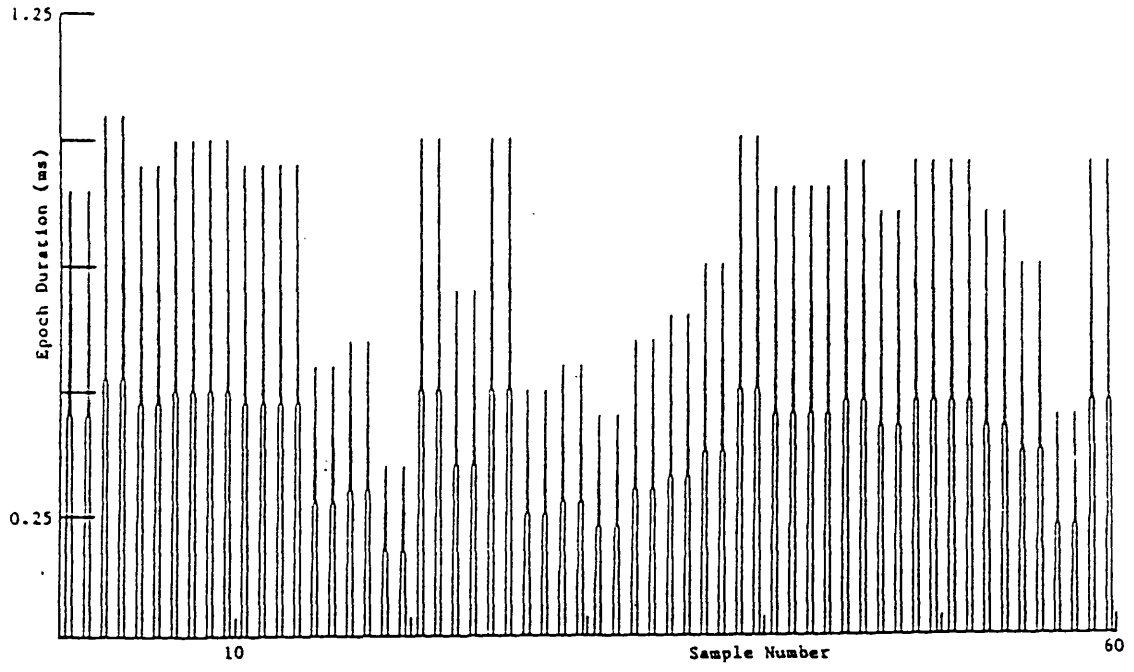


Figure 6.7 : (a) Sequence of Figure 6.3(a) after Forward and Inverse Hadamard transformation (N = 16) employing Low Pass Sequency Filtering of the transform coefficients.

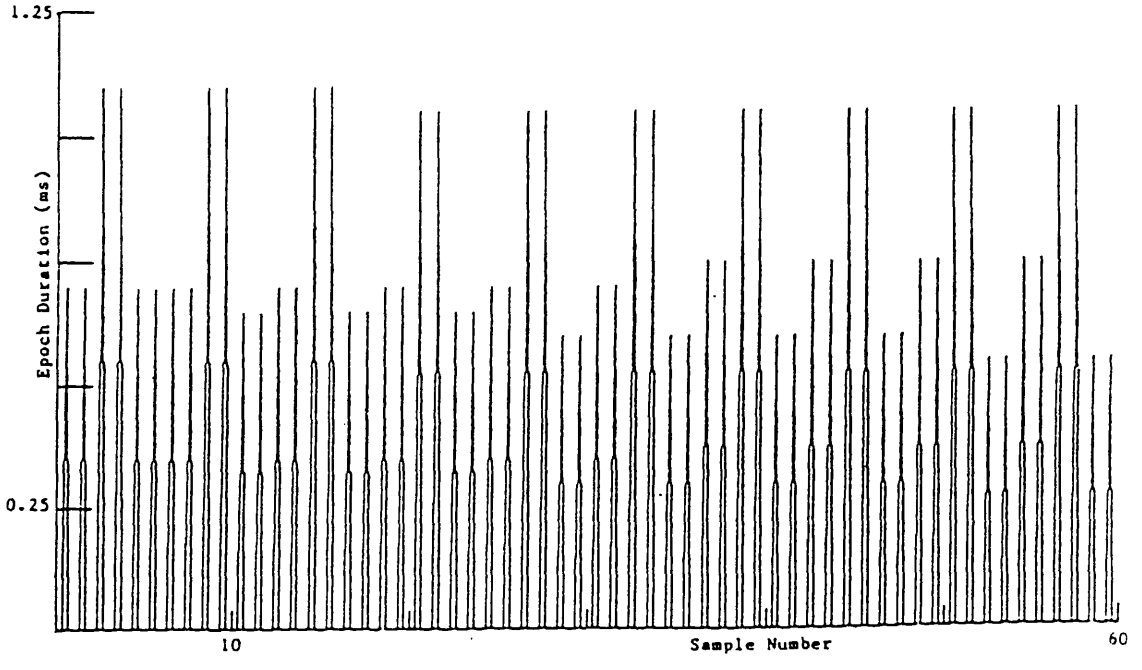


Figure 6.7 : (b) Sequence of Figure 6.3(b) after Forward and Inverse Hadamard transformation (N = 16) employing Low Pass Sequency Filtering of the transform coefficients.

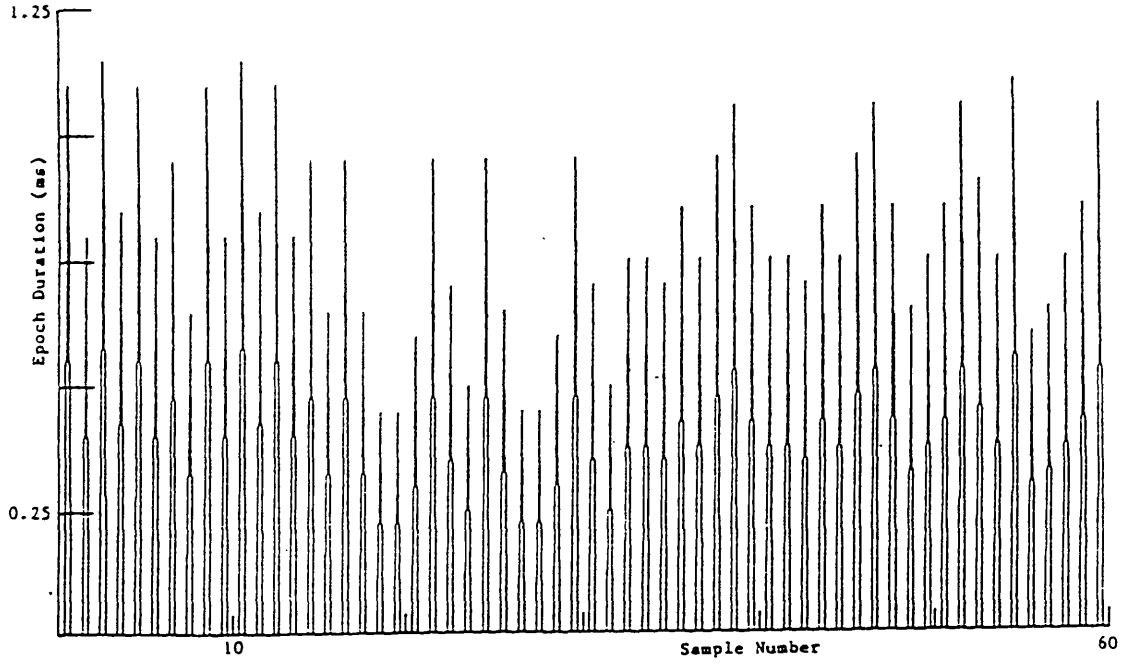


Figure 6.8 : (a) Sequence of Figure 6.3(a) after Forward and Inverse Hadamard transformation (N = 16) employing Multiple Bandpass Sequency Filtering of the transform coefficients.

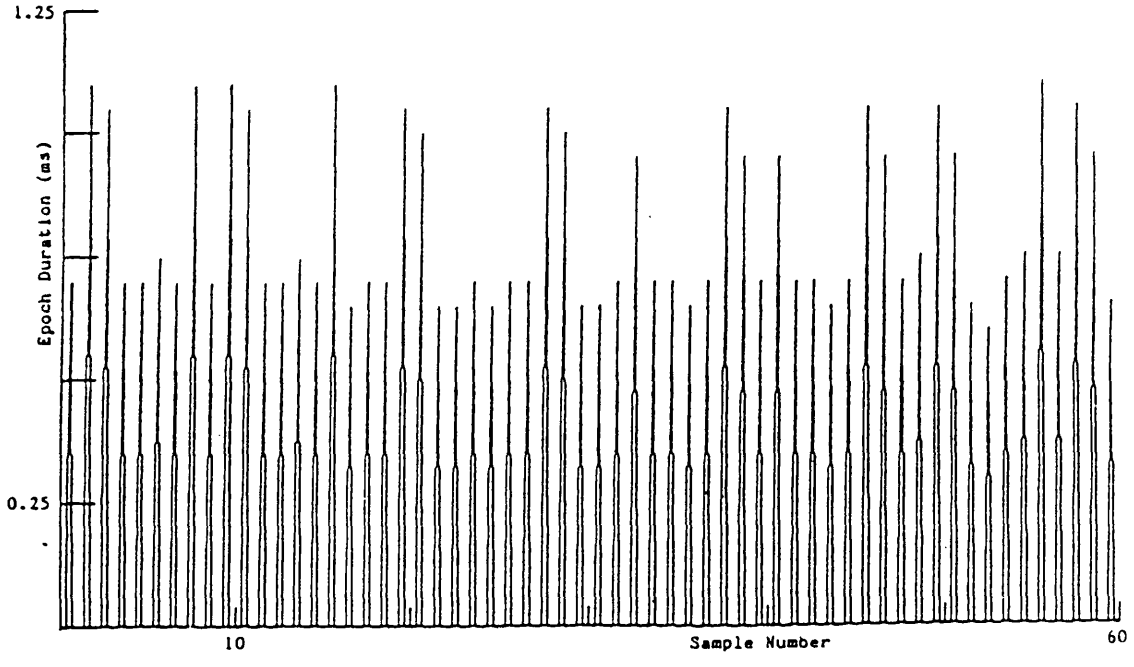


Figure 6.8 : (b) Sequence of Figure 6.3(b) after Forward and Inverse Hadamard transformation (N = 16) employing Multiple Bandpass Sequency Filtering of the transform coefficients.

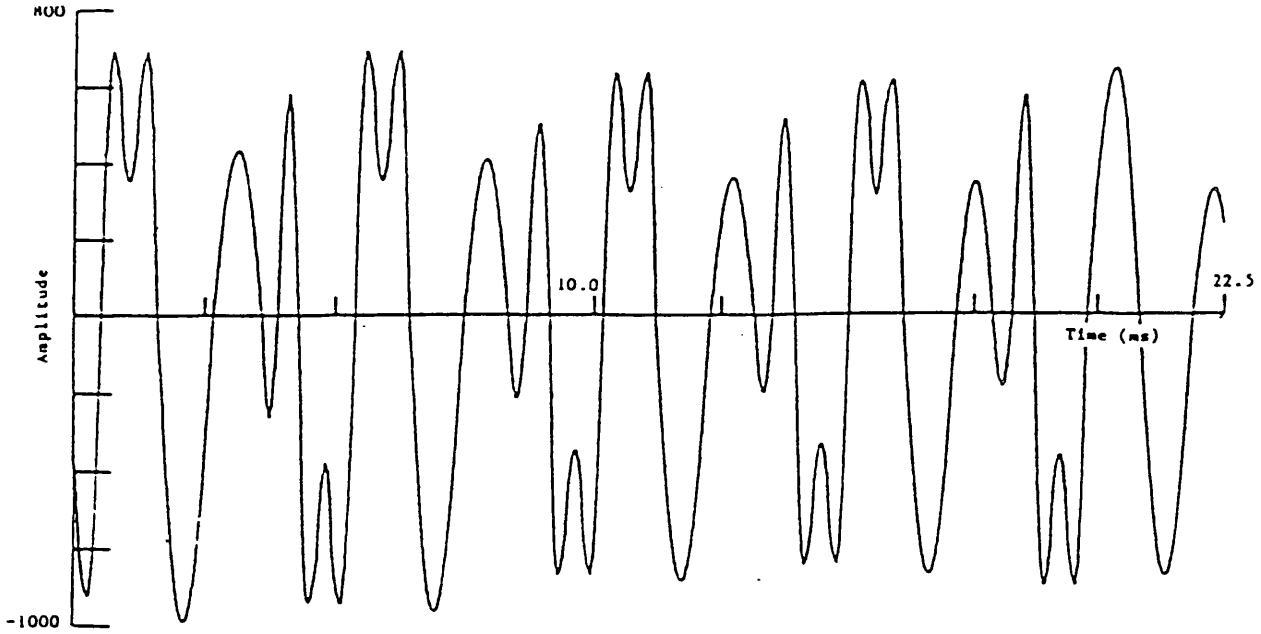
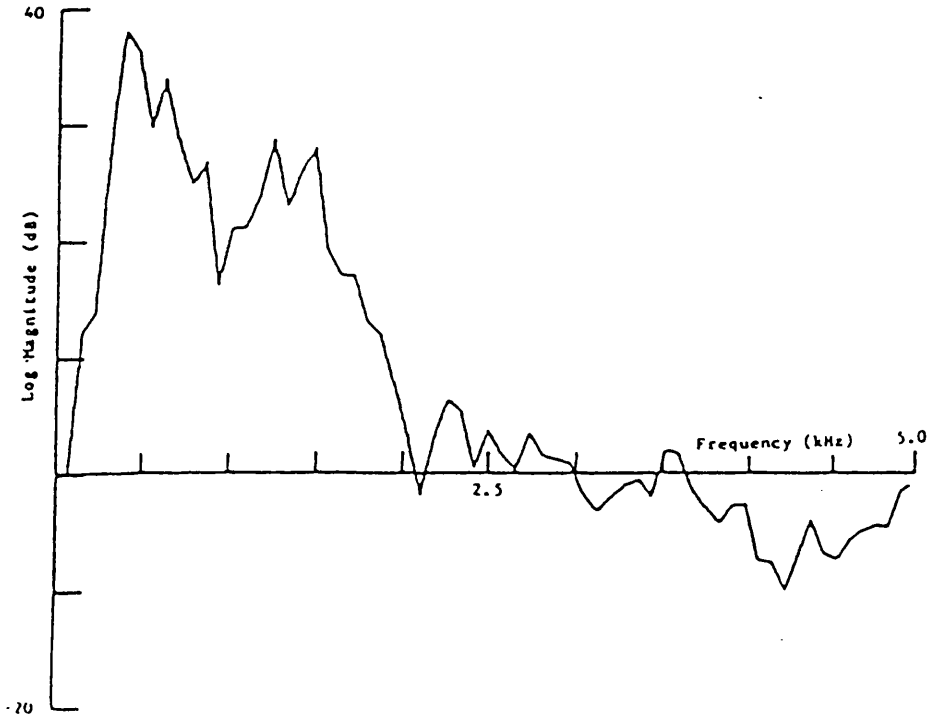


Figure 6.9 : (a) Segment of speech waveform synthesised using the input epoch duration sequence of Figure 6.3(b).

(b) Corresponding power spectral density plot.



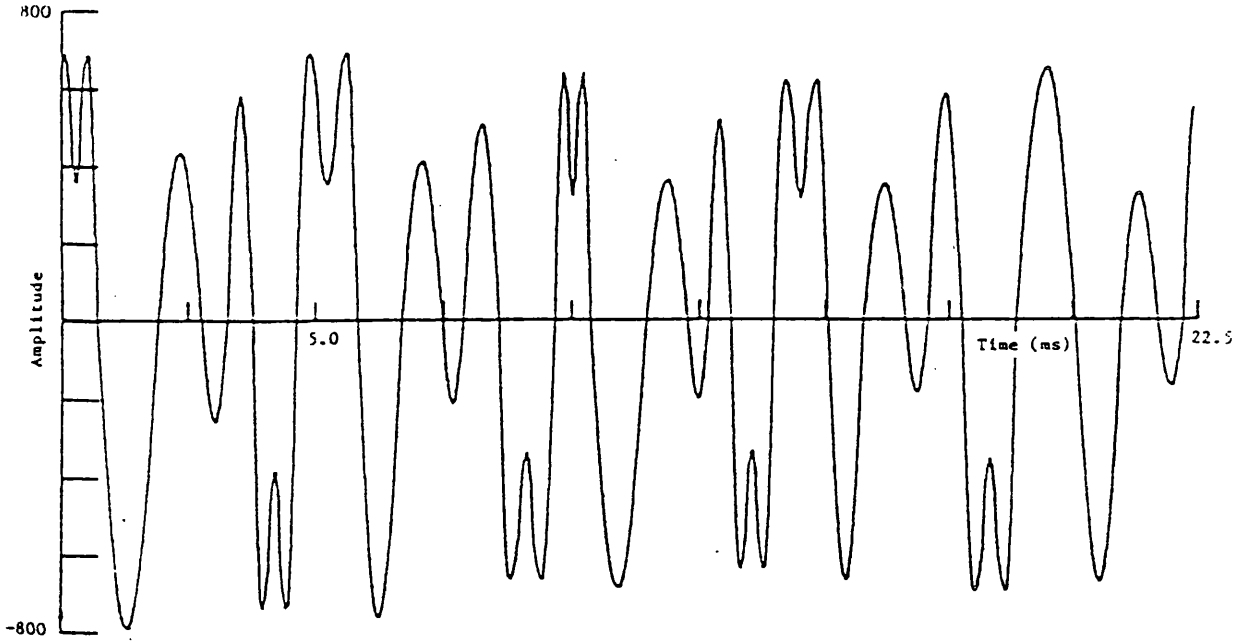
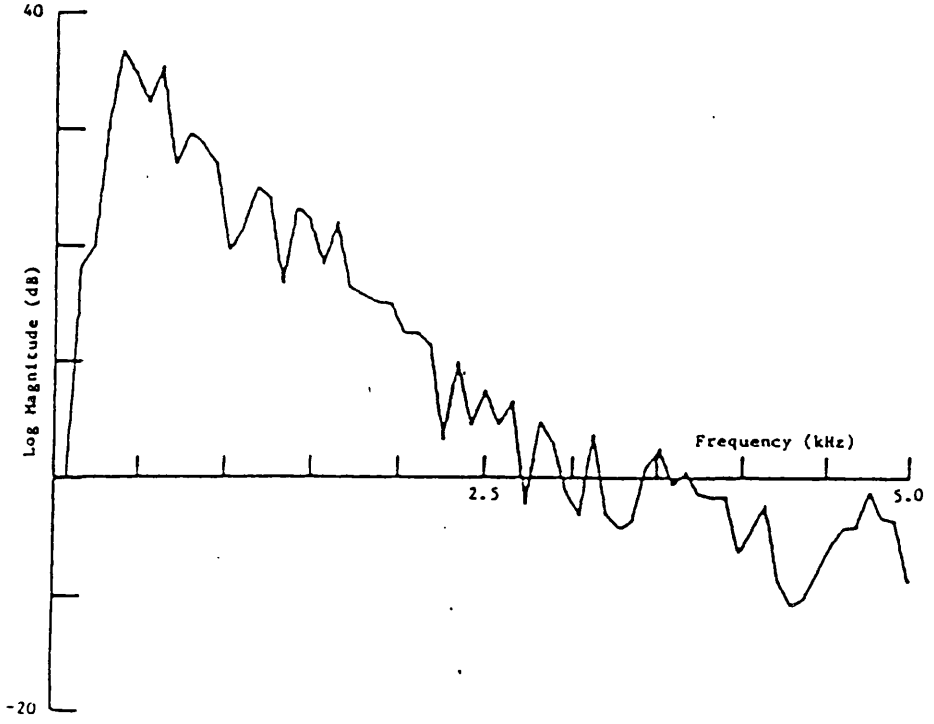


Figure 6.10 : (a) Segment of speech waveform synthesised using the epoch duration sequence of Figure 6.4(b).

(b) Corresponding power spectral density plot.



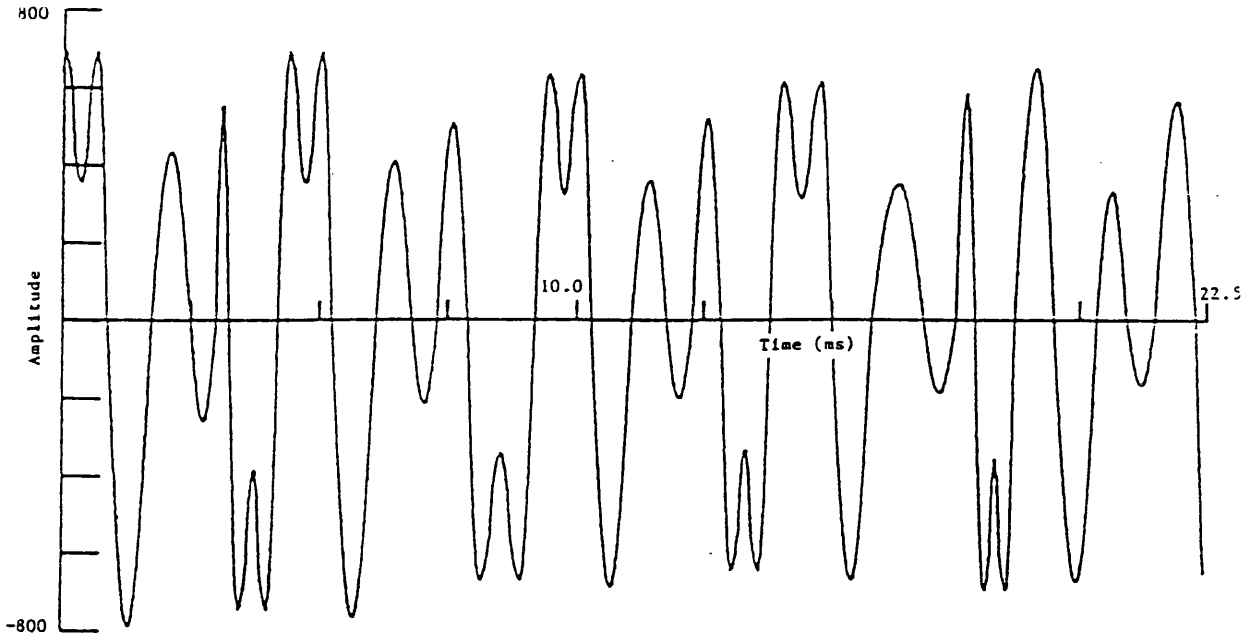
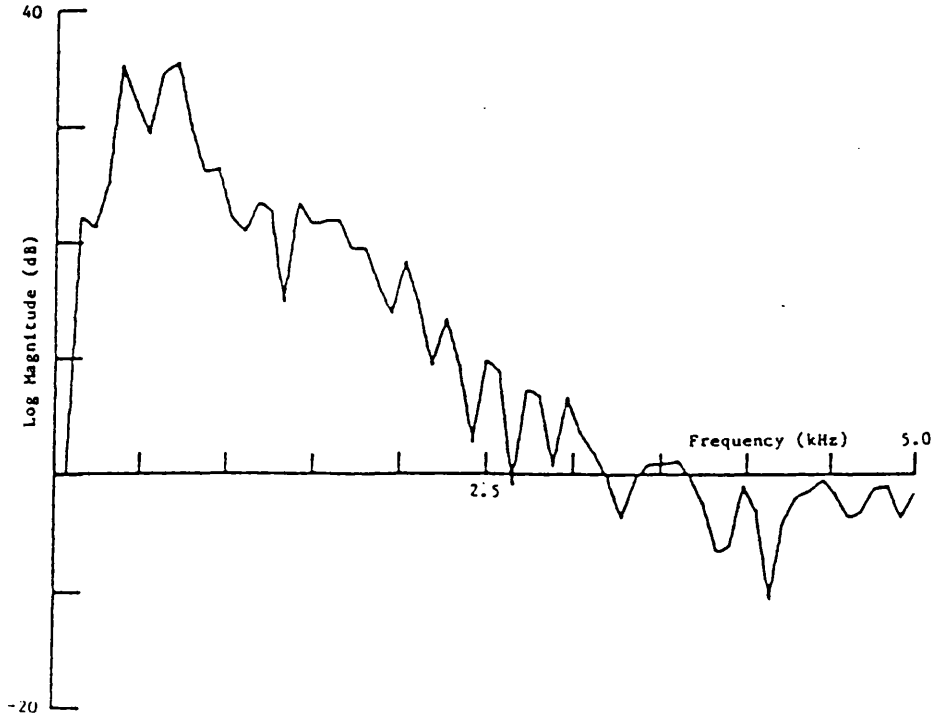


Figure 6.11 : (a) Segment of speech waveform synthesised using the epoch duration sequence of Figure 6.5(b).

(b) Corresponding power spectral density plot.



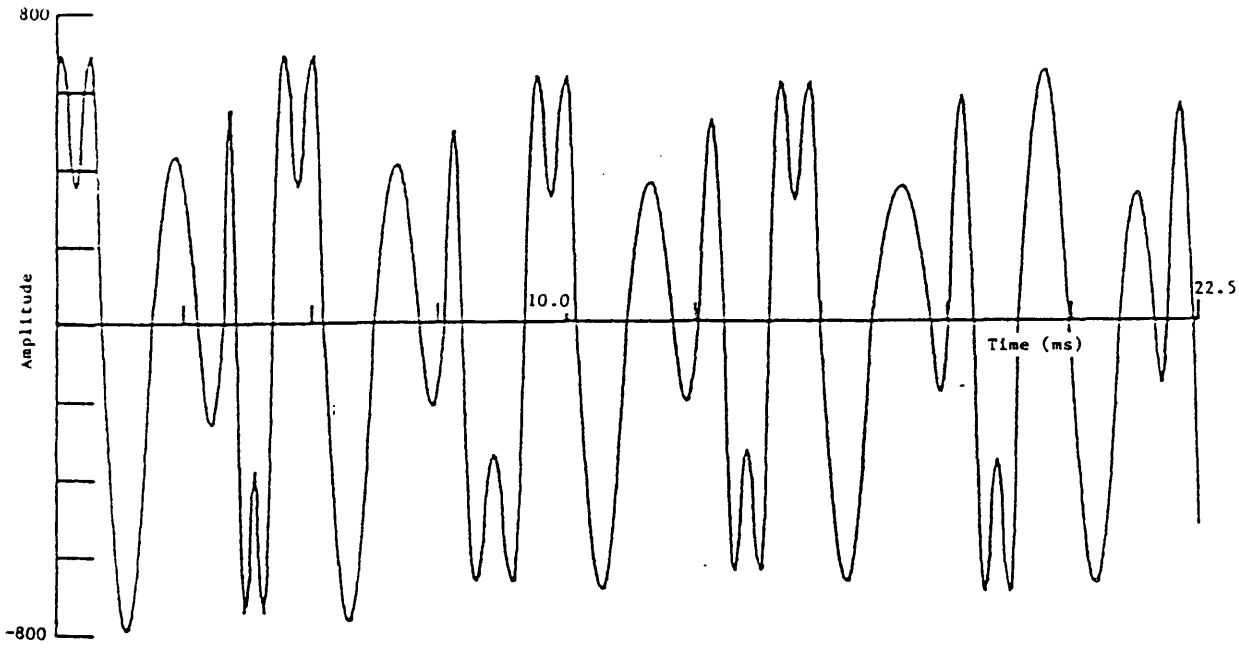
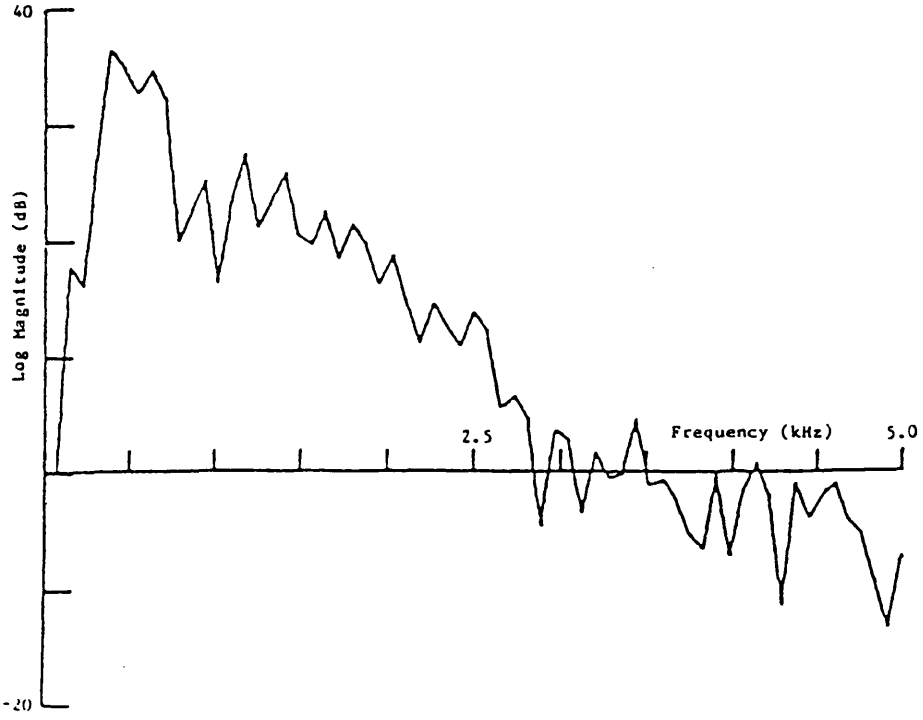


Figure 6.12 : (a) Segment of speech waveform synthesised using the epoch duration sequence of Figure 6.6(b).

(b) Corresponding power spectral density plot.



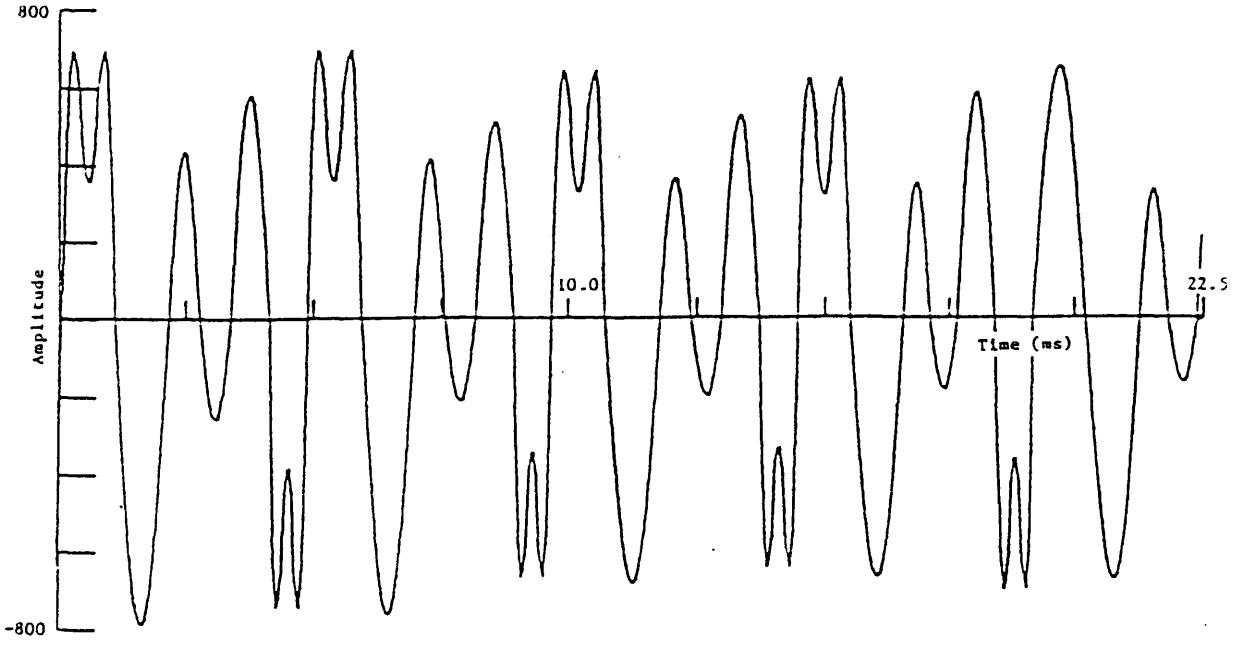
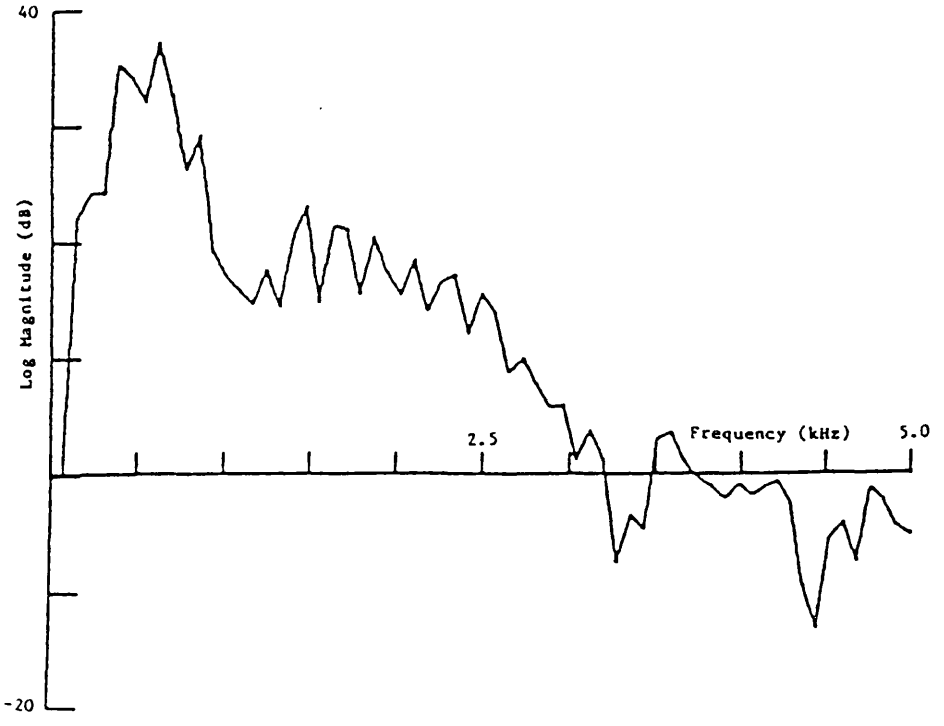


Figure 6.13 : (a) Segment of speech waveform synthesised using the epoch duration sequence of Figure 6.7(b).

(b) Corresponding power spectral density plot.



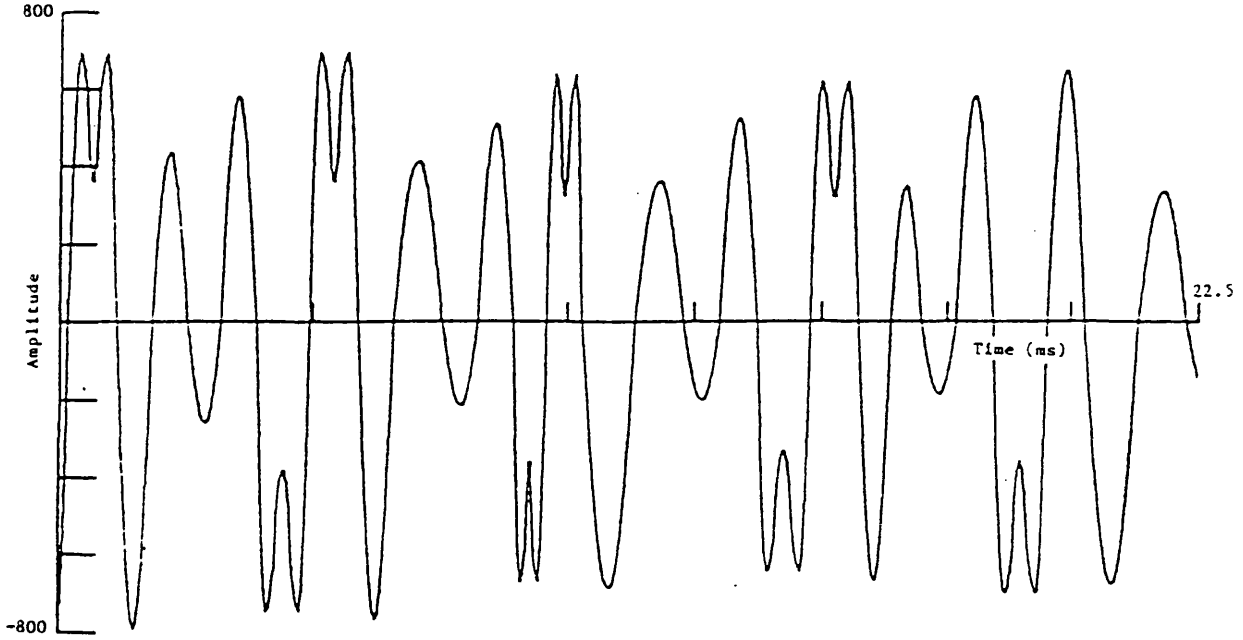
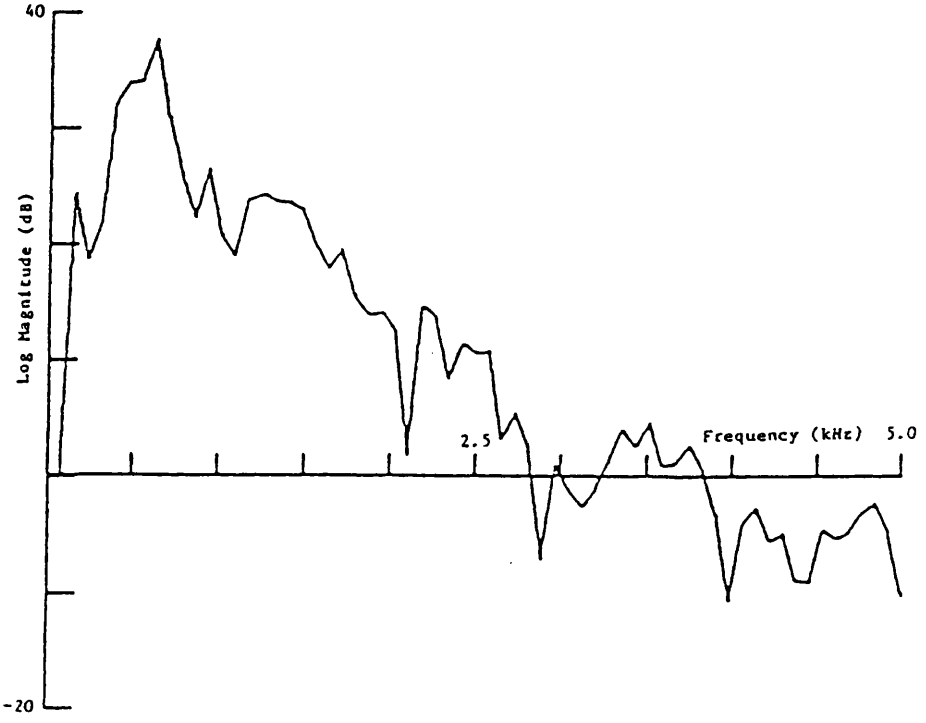


Figure 6.14 : (a) Segment of speech waveform synthesised using the epoch duration sequence of Figure 6.8(b).

(b) Corresponding power spectral density plot.



CHAPTER 7

Real-Time Implementations of TES and TES related Systems

7.1 Introduction

The development of a versatile real-time simplex digital voice channel was presented in section 2.3 of chapter 2. This system was developed for the real-time implementation and investigation of speech coding systems, and in particular TIME ENCODED SPEECH (TES).

This chapter highlights the implications of embodying TES and TES related algorithms upon the real-time simplex digital voice link for various bit-rates, system delays and data reduction techniques.

Section 7.2 reviews King and Goslings simulation algorithm and relates its real-time implementation to the algorithm developed for the real-time simplex digital voice channel. Al-Doubooni's simulation algorithm is reviewed in section 7.3. Also presented are the restrictions imposed/alterations required upon the coder before a real-time implementation was possible.

The system considerations for the implementation of the real-time coders are presented in section 7.4. The system parameters and a discussion of the implications of the real-time implementations are also included in this section.

Section 7.5 describes the TES coders which have incorporated either the differential or group amplitude signalling techniques investigated in Chapter 3.

The real-time TES coders with Hadamard Transformation of the epoch sequence are presented in section 7.6. The technique employed was investigated in chapter 6.

Section 7.7 presents an informal assessment of the coders developed and highlights some of the issues related to quality. Finally, section 7.8 presents the summary of the chapter.

7.2 TES Coder : King and Gosling

The Time Encoded Speech (TES) originally reported by King and Gosling utilised a coding alphabet of 23 coded shape descriptors [30]. Each data frame of the TES coder comprised of eight "coded shape descriptors" (tes-codewords) and a single eight bit codeword to communicate the mean amplitude of the preceding eight codewords. The waveshapes utilsed for the synthesis of the speech signal were square waves. Further research by King and Holbeche [93] demonstraated that a coding alphabet of 28 tes-codewords produced speech of superior quality to that utilising 23 tes-codewords. These investigations also inspected the utilisation of "rounded" waveshape for synthesis. However, the quality of the speech synthesised was judged to be poorer than that synthesised for square wave synthesis waveshapes.

From the description of the digital voice channel it is observed that the software for the channel does not require any alterations for the implementation of King and Goslings TES coder. The software support enabled a coding alphabet of 28 tes-codewords to be formed. The synthesis waveshapes of the alphabet are presented in figure 7.1. Inspection of figure 7.1 reveals that, for some of the multiple extrema segments, an element of skewness was incorportated. Informal appraisal of speech synthesised from coding alphabets with

varying degrees of skewness indicated that, subjectively, skewness towards the right most extreme yielded higher quality. The philosophy of King and Gosling was that epoch durations greater than 2ms (40 samples at 20kHz sampling) occur due to low frequency components and may be clasified as noise. Therefore, the synthesis waveshape for codeword 28 is shown to be a square wave of 40 samples. However, this waveshape, when implemented in a TES coder by King and Gosling, had zero amplitude. Figure 7.2 shows the coding matrix for the 28 codeword alphabet. This diagram also highlights the quasi-logarithmic quantisation of the epoch durations described by King and Gosling. The exception to this is the epoch duration of 0.05ms (1 sample) which was signalled as an epoch of 0.1ms (2 samples). The codeword numbers of figure 7.2 are positioned at the synthesis "co-ordinates". For example, when an epoch of 0.75ms duration and 3 minima is detected codeword 12 is transmitted. On receiving this codeword the receiver synthesised an epoch of 0.85ms duration with 2 minima.

7.3 TES Coder : Al-Doubooni

Al-Doubooni's TES coder [39] differed from that originally developed by King and Gosling. As described in chapter 2, mapping of the quantised time and shape descriptors onto a reduced alphabet was excluded and the amplitude information was communicated on an epoch to epoch basis, rather than after every eight epochs. To reduce the effects of background noise and the data required for encoding periods of silence, symmetrical thresholding of the speech signal was applied. Epochs with a peak magnitude less than the threshold level

were assigned a zero valued peak magnitude. A succession of such epochs were combined to form a single epoch of zero magnitude and duration equivalent to the sum of the durations of those epochs combined.

The epoch parameters 'transmitted' were : Epoch duration, Number of minima and peak magnitude. Al-Doubooni employed the full quantisation range for peak magnitude (0 to 511) and epoch duration (1 to 2047 samples) encoding. Extra codewords were required to signal the extrema information.

To implement such a coder, the real-time channel software required alterations to accommodate a data frame comprised of one amplitude codeword and one tes-codeword. Although Al-Doubooni transmitted extrema information when synthesising the speech signal a half waveform with either one or three extrema was used. Thus, in the real-time system only two stylised waveshapes were required. These are given in figure 7.3.

Due to the symmetrical thresholding, epoch durations of the order of 0.1ms (2000 samples) were possible in the simulation algorithm. To encode epochs of such duration in a real-time system would involve excessively large alphabets and system delay (since the minimum system delay must be equal to the largest epoch duration). To overcome this, all epoch durations between 0.05 and 1.95ms (1 to 39 samples) inclusive were considered valid. Any single epochs of a duration greater than 1.95ms (39 samples) were classified as silence and represented by a single codeword representing an epoch of 2.0ms duration (40 samples) and zero amplitude. To preserve polarity, the

number of epochs combined by the symmetric thresholding was restricted to odd values. When the sum of the epoch durations of the combined epochs, T_c, was greater than 1.95ms, a codeword representing an epoch of 2.0ms and zero amplitude was transmitted and T_c was set equal to T_c - 2.0ms. Thus, time warping was not introduced by this technique and the excessively large epochs produced by Al-Doubooni's simulation algorithm did not occur.

In the simulation algorithm the number of extrema were signalled for all epoch durations. However, epoch durations of 0.05 and 0.1ms (1 and 2 samples, respectively) may only be represented by half waveforms with one extreme. Since the synthesis routine required only one of two waveshapes the represention of extrema information required only one bit codeword.

The digital voice link of chapter 2 was developed for the transmission of two independant codewords. Significant software alterations would have been necessary for the transmission of amplitude information, epoch duration and the number of extrema. It was therefore chosen to combine the extrema and epoch duration information to prevent the need for software alterations. To represent all valid epochs 2 codes were required for the represention of epoch durations of 0.05 and 0.1ms, each of the epochs in the duration range of 0.15 to 1.95ms inclusive, may be represented by 2 codewords per epoch (either a single or multiple extrema synthesis waveform) and one codeword to represent epoch duration of 0.2ms was required. Therefore, to implement a real-time equivalent of Al-Doubooni's TES coder an alphabet of 77 tes-codewords was required for the coding

matrix. This is given in figure 7.4. As in Figure 7.2 the codeword numbers of figure 7.4 are positioned at the synthesis co-ordinates.

7.4 System Considerations

The performance of a real-time digital voice transmission system depends upon transmission rate, length of transmitter (Tx.) and receiver (Rx.) buffers and the overall system delay introduced by the buffer arrangment. Operating at modest transmission rates with finite buffer lengths and system delay, which enable effective two way communications, will inevitably introduce buffer overflowing and underflowing, causing further distortions which degrade the speech quality.

(i) Buffering

Figures 7.5(a) and (b) illustrate the variations in source symbol generation rate for two utterances possessing widely different characteristics. To produce these figures the symbol rate was defined over a period of 5ms and plotted against time. The speech signal had a bandwidth of 0.1 to 4.5kHz.

These utterances were characterised by an approximately uniform mean symbol generation rate in the range 0.5 to 1.5 ksymbols/second for the majority of the time, which corresponds mainly to voiced sounds, while sustained bursts of high generation rate occurred due to unvoiced sounds. Constant rate transmission of the symbols therefore required buffering to smooth out the fluctuations in

generation rate.

The buffer structure employed in the simplex digital voice channel was such the transmitter (Tx.) and receiver (Rx.) buffer lengths were defined by the maximum number of codewords the buffers were capable of storing, and not the maximum number of bits. Therefore, if the buffers employed in each coder were of equal lengths (in the sense of the number of codewords they were capable of storing) then the maximum number of data frames which may be stored in Al-Doubooni's coder (hereafter referred to as ADcoder) will be greater than in King and Gosling's coder (hereafter referred to as KGcoder). However, the data frames stored in the buffers of the KGcoder represent more epochs than the data frames stored in the buffers of the ADcoder. Using the same argument, if the length of the buffers were specified in bits then again the ADcoder would be capable of storing more data frames than the KGcoder but, the data frames stored in the KGcoder would represent a greater number of epochs and therefore a greater period of time (segment of speech).

Since buffers of the same length (whether defined in terms of data frames, codewords or bits) were capable of storing codewords which represent segments of speech of differing lengths, so the rates at which the Tx. buffers filled, for the same segment of speech, also differed. This implied that the point at which Tx. buffer overflow became imminent would differ in the ADcoder and KGcoder. Although the mechanisms for Tx. buffer overflow managment were identical in each coder, its interaction occurred at different points within the speech signal. Therefore, the auditory effects would be dissimilar.

As stated earlier, the data frames employed by Al-Doubooni differed from that originally specified by King and Gosling. To reiterate, Al-Doubooni structured a data frame employing one 9 bit amplitude codeword and one 7 bit tes-codeword (Coded Shape Descriptor), a total of 16 bits per data frame. King and Gosling utilised one 8 bit amplitude-codeword and eight 5 bit tes-codewords, a total of 48 bits per data frame.

If N data frames were to be discarded to prevent buffer overflow, then a total of 16.N bits (N tes-codewords) in the ADcoder, or 48.N bits (8.N tes-codewords) in the KGcoder, would be discarded. To discard a set number of data frames for the prevention of Tx. buffer overflow results in a greater number of tes-codewords being discarded in the KGcoder than in the ADcoder. Therefore, a larger segment of speech is discarded by the KGcoder than the ADcoder. If the discard were conducted in terms of bits then, 3 data frames of the ADcoder must be discarded for each data frame discarded by the KGcoder. However, once again more epochs are discarded by the KGcoder than the ADcoder.

From the above it was clear that, even in the ideal situation of distortionless transmission within both coders, for the storage of either an equal number of codewords or data frames within each coder buffers of different lengths were required in each coder. This situation occurred because the structure of the buffers in the digital voice channel were word-storage orientated. To specify buffers of different lengths in each coder implied different durations of transmission delay.

Only if the buffer lengths were defined in bits would each coder have identical values of system delay. However, in this situation they had different characteristics since they were capable of storing different quantities of dataframes/codewords.

The implementation of the coders was to be bit orientated since each coder was to be studied using various serial bit transmission rates. It was therefore decided to specify the length of the buffers and the extent of the data discarded/repeated to prevent Tx. buffer overflow/underflow, in bits. However, due to differing lengths of codewords in each coder identical values of Tx. and Rx. buffer lengths could rarely be achieved.

(ii) Transmission Delay

Time Encoded Speech involves the coding of variable-rate source signals into constant-rate signals for transmission. Buffer storage at transmitter and/or receiver is required to permit this "variable-rate to constant-rate" transformation. As a result, delays are introduced into the communication process. This form of delay is commonly referred to as Transmission Delay.

The mechanisms involved, the influences of buffer size and the delays associated with TES systems were described by Turner et al [34], who concluded that "fairly long delays (of the order of two seconds round trip delay) may be involved in the distortionless transmission of speech encoded using information about the waveform segments linking successive real zeros from speech signals". Extending

this analysis Mason and Balston [35] proposed a modest increase (by a factor of 1.5 to 1.9) in transmission rate over the average source generation rate for the distortionless transmission of time encoded speech with a tolerable transmission delay of 200ms (400ms round trip delay). Turner et al [36] derived explicit expressions for the limiting delay in time encoded speech type systems operating at data transmission rates higher than the average rate at which the information is produced by the variable-rate source. However, King and Holbeche [37] offered experimental evidence which indicated that transmission delays in time encoded speech may be reduced an order of magnitude by utilising sub-optimum bounded entropic codes in place of constant length codes.

A number of researchers have studied the effects of delay upon voice transmission. Krauss and Bricker [94] had male and female subjects converse and solve puzzles over circuits with round trip delays of zero, 600ms and 1800ms. Measurements of word counts and subjective opinions indicated no significant differences between zero and 600ms delays, but the 1800ms delay caused an increase in reports of "difficulty in communicating due to the circuit".

Klemmer [95] found that users were very seldom disturbed by delays of 600ms and 1200ms. These studies appeared to indicate that subjects were generally unaware of, and undisturbed by, round trip delays of upto 1200ms.

Transmission delay is a highly contentious subject in the field of communications. The contention is, in the main, due to the techniques utilised for the testing of telephonic systems. The majority

of results presented are based upon "subjective opinions".

The previous discussion of buffering highlighted the fact that the codewords in the two coders were of different bit lengths and that the number of codewords which constituted a data frame were not always the same. Since the buffers of the coders were word-storage orientated the buffer lengths could not be precisely specified in bits. Instead they were specified to a length which enabled the storage of an integer number of codewords.

For many data frames and codewords implemented it was impossible to define the transmission delay precisely in each coder. Therefore, the delay nearest to 300ms (or whatever value) was utilised.

To implement the real-time algorithms presented in this thesis, upon the simplex digital voice channel, it was chosen to employ a transmission delay of the order 300ms (equivalent to round trip delay of the order of 600ms) since the majority of researchers dispute the results obtained for delays greater than this figure.

(iii) Transmission Rate

The subject of digital speech encoding and bit-rate compression has been one of considerable interest in recent years. Attention has focused strongly on bit rates in the range of 9.6 to 16kb/s for applications where good "communications quality" and robustness across a wide range of background noise conditions and speaker variations is required [96]. The bit rate of 9.6kb/s appears, at present, to be

about the lowest practical rate at which "communication quality" and robustness can be readily achieved. Below 9.6kb/s presently known techniques have a noticable synthetic quality and are considered more fragile to differences in speakers and background conditions [96].

Although this thesis investigated data reduction techniques for TES it has not strived to develop the ultimate TES coder. In fact, the real-time implementations of Al-Doubooni and King and Gosling coders still require extensive research before they may be claimed to be in any sense the "optimum" TES coder.

The MIPROC system, upon which the real-time digital voice channel was implemented, has several pre-set clock signal generators which may be employed for transmission timing. The pre-set clock rates are 5, 10, 15 and 20kb/s.

Since the algorithms implemented in Real-time were not "optimum" it was chosen to employ the pre-set clock rates of 10 and 15kb/s for the transmission timing. In some instances a transmission rate of 4.8kb/s was inspected.

7.4.2 System Parameters

Table 7.1 presents the system parameters utilised for the real-time implementation of Al-Doubooni's and King and Goslings coder.

This table highlights the differences in the size of data frame employed in these coders. Because the data frames were of different

sizes so the filler-words, which prevent Tx. buffer underflow, also differed since each filler-word had to have the same number of bits as the complete data frame.

The extrema sensitivity parameter of the extrema detection algorithm was set at 44dB below the maximum possible peak magnitude, in all the coders developed. This parameter, which is described in chapter 2, eliminated the very small extrema generated by either low level noise or last bit uncertainty within the analogue-to-digital converter.

7.4.2 Discussion

From figures 7.5(a) and (b) it is observed that the rate of generation of epochs for undifferentiated bandlimited speech may vary from as low as 200 epochs/second, which results from voiced segments of speech, upto 8000 epochs/second which occurs due to bursts of unvoiced speech. The speech employed to produce the plots of figures 7.5(a) and (b) was bandpass filtered from 0.1 to 4.5kHz. To impose the telephone bandwidth of 0.3 to 3.4kHz would reduce the upper generation rate to approximately 5000 epochs/second. Conversely, the lower generation rate would be increased to approximately 800 epochs/second.

To signal epoch information the ADcoder utilised 16 bits per epoch. Therefore, the rate at which symbols (bits) were generated varied from 12.8kb/s upto 80kb/s. Clearly, with transmission (Tx.) buffer delays of the order of 100ms and serial transmission rates of

10 or 15kb/s, Tx. buffer overflow was a frequent event in the ADcoder. Increasing the system buffer delay reduced the extent of buffer overflow. However, distortion began to arise due to the buffer managment dicarding/repeating technique. As the length of the Tx. buffer was increased the condition arose where the Tx. buffer was able to store a small segment of voiced speech and a complete segment of unvoiced speech. At this point the Tx. buffer began to overflow and data was discarded. Because of the discarding of data by the Tx. buffer the Rx. buffer began to underflow. The last M data frames within the Rx. buffer were therefore repeated to prevent total Rx. buffer underflow. However, the repeated segment represented a voiced segment of speech and the repetition of such a segment produced a grossly distorted waveform.

The situation also arose where the TES codes repeated represented a segment of inter-word silence. The repetition of such segment also resulted in a significant loss of input speech.

The KGcoder employed 48 bits to signal information for a group of eight epochs. Therefore, the symbol generation rate for this coder varied from 4.8kb/s upto 30kb/s. The extent of Tx. buffer overflow was significantly less than that of the ADcoder for the same parameters. If the system delay was increased such that the occurrences of Tx. buffer overflow were reduced until a condition of distortionless transmission was achieved, the audible effects encountered with the ADcoder did not occur.

7.5 Amplitude Signalling

The results of the investigations reported in chapter 3 indicated that amplitude information of the order of one bit per epoch produced speech of acceptable quality. The speech of highest intelligibility and acceptability was produced utilising the following techniques:

(a) Employing 2 bit Differential Pulse Code Modulation (DPCM) for successive epoch amplitude parameters.

(b) Signalling the amplitude parameter for a group of N epochs.

Implementation of the 2 bit DPCM of the epoch peak amplitude (hereafter referred to as amplitude) required substantial software changes within the analysis and synthesis routines of the real-time algorithms. The alterations required within the ADcoder were relatively straight forward and a data frame consisted of a 2 bit amplitude-codeword and a 7 bit tes-codeword. To retain the data frame structure of the KGcoder would have significantly increased coder complexity and therefore reduced the viability of TES. It was therefore chosen to employ the same data frame structure as utilised by the ADcoder. Thus a data frame consisted of a 2 bit amplitude-codeword and a 5 bit tes-codeword.

The signalling of the amplitude parameters for groups of epochs involved detection, over N successive epochs, of the maximum epoch amplitude as well as the individual epochs amplitudes. Each epoch within the group was normalised such that the individual epochs

amplitudes were equal to that of the maximum amplitude of the group.

The signalling of amplitude information over groups of epochs was precisely what King and Gosling originally proposed. The only difference being the group size. In the investigations of chapter 3 an 11 bit amplitude-codeword was employed for signalling the amplitude of 11 epochs. King and Gosling utilised an 8 bit amplitude-codeword for a group of 8 epochs. However, the intrinsic information differs. To employ an 8 bit codeword, instead of an 11 bit codeword, results in the amplitude information being coarsly quantised which increases the quantisation noise. However, to code a group of eight epochs instead of eleven reduces the extent of the amplitude distortions introduced by normalisation of each epoch within the group to the maximum value within the group. The investigation reported in chapter 3 indicated that it was difficult, but not impossible, to discriminate between algorithms using groups of 8 and 11 epochs. However, in those investigations a group size of eight was equivalent to 1.37 bits of amplitude information per epoch.

The algorithms which incorporated amplitude signalling over groups of N epochs were developed from the original KGcoder. Clearly, no alterations were required for the inclusion of this technique within the KGcoder. However, a threshold level was incorporated within another version of this algorithm such that, if the maximum amplitude of the group of epochs, A_N, was less than the threshold level, then A_N was set equal to zero.

The data frames employed were: (a) one 8 bit amplitude-codeword and eight 7 bit tes-codeword (for the ADcoder) and one 8 bit amplitude-

codeword and eight 5 bit tes-codeword (for the KGcoder).

7.5.1 System Parameters

The system parameters for the implementation of the ADcoder and KGcoder with amplitude information signalled for groups of eight epochs are presented in Table 7.2. From comparison of the parameters for the coders employing the King and Holbeche coding alphabet (KHalphabet) of Tables 7.1 and 7.2 it is observed that they differ only in the number of data frames manipulated by the buffer managers.

Although the number of codewords forming a data frame were the same for each coder the number of bits per data frame differed. Therefore, to ensure that the filler-word was unambiguously decoded by the receiver of each coder, the filler-word was formed from four 16 bit words. Since the number of bits forming a data frame differed so the number of data frames discarded/repeated by the buffer managment software had to differ to ensure that the same quantities of data were being discarded by each coder.

The system parameters for the coder which incorporated differential amplitude signalling are given in Table 7.3 which shows that, for this particular coder, a filler-word of only 16 bits was required. Due to the differing sizes of data frame 3 data frames were discarded by the coder when Al-Doubooni's coding alphabet (ADalphabet) was implemented compared with 4 data frames when KHalphabet had been incorporated.

7.5.2 Discussion

The algorithms which employed amplitude signalling over groups of 8 epochs yielded very similar bit generation rates to that of the KGcoder of Table 7.1. The extent of Tx. buffer overflow experienced with the original ADcoder was significantly reduced in the version of Table 7.4 because the information employed for signalling epoch amplitude had been reduced from nine to one bit per epoch. However, the ADcoder of Table 7.2 yielded a greater bit generation rate than that of the KGcoder.

The buffer managment technique of discarding and repeating data frames created its own problems in the differential amplitude signalling implementation. The differential step sizes were set at 42dB and 20dB below the maximum possible peak amplitude. When an imminent Tx. buffer overflow condition occurred, the buffer manager discarded N data frames. At some time after this event the Rx. buffer reached a condition of imminent underflow and the buffer manager repeated the last N data frames. This technique had been sufficient for the coders described so far. However, since this coder utilised differential amplitude signalling, the discard and repeat of the codewords destroyed the amplitude sequence. This resulted in the incorrect levels being synthesised and the speech synthesised was noisy and unintelligible. If left sufficiently long enough, the system occasionally recovered from this situation until the next discard/repeat.

This distortion was considered unacceptable and measures were required to prevent this or to ensure a quick recovery. One tech-

nique would be to encode the amplitude information prior to transmission. This would ensure that the discarding of data would not affect the amplitude signalling. However, the point in time, after the Tx. buffer discard, at which the Rx. buffer repeats a segment cannot be predicted. Therefore, the data transmitted cannot be appropriately adjusted to prevent a discontinuity of amplitude levels caused by the repetition at the receiver.

It was chosen to employ a technique which ensured a quick recovery from the effects of buffer overflow rather than attempt to adapt the codes at such times. To do this, four different step sizes were specified for the coder such that, the rate of decay of the synthesised speech was much greater than the rate of attack. The step sizes for increasing the amplitude were 42dB and 20dB below the maximum possible peak amplitude and the step sizes for decreasing the amplitude were 36dB and 14dB below the maximum possible peak amplitude.

When using these step sizes, if a discard/repeat was performed, which totally corrupted the sequence for decoding, resulting in unintelligible speech, it was found that the speech level was quickly forced to zero eliminating the disturbing audiable effects. The algorithm also tended to recover quickly from this situation where if the standard algorithm managed to recover, in general, it a significant period of time had elapsed before recovery.

This technique had two advantages. Firstly, instead of the listener having to tolerate, sometimes quite loud, noise the level was rapidly forced to zero. This produced a significant perceptual

effect upon the speech quality and intelligibility. Secondly, the algorithm recovered from the disturbances introduced by the discard/ repeat mechanism.

7.6 Orthogonal Transformations

A maximum data reduction of 1.35:1 was achieved employing Dominant Coefficient Retention within a Hadamard Transform (N = 16). The resulting speech was intelligible but of moderate quality.

The speech synthesised from the Low-pass Sequency Filtered Hadamard coefficients was of very good quality and intelligibility. A maximum data reduction of 1.33:1 was achieved for N = 4. However, analysis of this process (appendix 6) demonstrated that an identical epoch duration sequence (to that output from the transform) was achieved if the average of pairs of adjacent epoch durations was calculated and the original epoch durations were replaced with the average value. This process yields a 2:1 data reduction in the epoch duration sequence because only the average value need be transmitted. This process also reduced the delay introduced by the transformation from N epoch durations to two epoch durations, where N was the size of the Hadamard transform. ie. by a factor of N/2.

The coded shape descriptors contain information concerning the epoch shape as well as duration. To implement the transformation of epoch duration only, within a real-time system, would have increased the data required for the representation of epoch parameters within the KGcoder. Significant software alterations and redesign of the

codeword alphabets would also have been necessary for such an implementation. It was therefore decided that the epoch duration and shape information (extrema count) were to be 'transformed' simultaneously.

The pseudo-Hadamard Transform (here after referred to as Hadamard Transformation only) yielded the greatest data reduction for the least system delay and complexity of implementation. This technique was therefore adopted rather than implementing a standard Hadamard Transformation within transmitter and reciever.

Two algorithms were developed for the inclusion of Hadamard Transformations of the epoch duration and extrema count. The first algorithm developed utilised the structure of the ADcoder. A data frame within this coder, the Hadamard Transform ADcoder (HTADcoder), consisted of two 9 bit amplitude-codewords and one 7 bit tes-codeword. To incorporate such a data frame the Receiver Interrupt Service and Analysis and Synthesis routines of the ADcoder were altered to develop the HTADcoder.

These alterations were necassary because the HTADcoder transmitted two consecutive amplitude-codewords per tes-codeword while the ADcoder transmitted one amplitude-codeword per tes-codeword. If the KHalphabet was utilised within the HTADcoder algorithm then the data frame consisted of two 8 bit amplitude-codewords and one 5 bit tes-codeword.

The second algorithm was developed from the KGcoder. Once again the majority of software changes were conducted within the

analysis and synthesis routines. However, a data frame within this coder, the Hadamard Transform KGcoder (HTKGcoder), was comprised of one 8 bit amplitude-codeword and four 5 bit tes-codewords. If the ADalphabet was incorporated within the HTKGcoder then one 8 bit amplitude-codeword and four 7 bit tes-codewords formed the data frame.

7.6.1 System Parameters

The system parameters for the HTADcoders implemented are presented in Table 7.4. The filler-word for both coders was 32 bits (two 16 bit words). The number of data frames discarded/repeated by the buffer manager were 4 (for the HTADcoder with ADalphabet) and 6 (for the HTADcoder with KHalphabet). This represents 100 and 126 bit discarded/repeated, respectively. As in the case of the coder of section 7.5, to discard an equal number of bit from each coder would have required extensive data to be discard which would have been impractical.

The HTKGcoder also employed a filler-word of 32 bits. These coders signalled amplitude information for groups of eight epochs. Comparing Tables 7.1 and 7.5 it is observed that the filler-word required by the HTKGcoder is half that for the KGcoder. The data required to form a data frame has been reduced by ratios of 1.5:1 and 1.7:1 when the coders include the ADalphabet and KGalphabet, respectively.

7.6.2 Discussion

The symbol generation rate for each of the algorithms of section 7.6 was less than that of the algorithms of section 7.2, irrespective of which coding alphabets were employed. This was a result of the reduced data requirement for each epoch. The HTADcoder with ADalphabet required 12.5 bits for signalling epoch information compared with 16 bits required by the ADcoder. If the KHalphabet was implemented in the HTADcoder then the bit requirement per epoch was further reduced to 10.5 bits.

The HTKGcoder with KHalphabet transmitted 28 bits per data frame which was equivalent to 3.5 bits per epoch compared with 48 bits per data frame (6 bits per epoch) for the KGcoder. If the ADalphabet was implemented then the data frame was 36 bits (4.5 bits per epoch).

Table 7.4 indicates that the prevention of Tx. buffer overflow/ Rx. buffer underflow in the HTADcoder involved the discarding/repeating of 4 data frames when the ADalphabet was utilised and 6 data frames for the HTADcoder with the KHalphabet. In the HTADcoder a data frame consisted of two amplitude-codewords and one tes-codeword. However, the tes-codeword represents two epochs and therefore the discard/repeat of 4 or 6 data frames actually involved 8 or 12 data framesof an ADcoder, repectively.

The HTADcoder (of Table 7.4) employ 10.5 bits per epoch (for the application of the KHalphabet) or 12.5 bits per epoch (for the application of the ADalphabet) compared with 16 bits in the ADcoder.

Therefore, for every data frame stored in the Tx. buffer of the ADcoder 1.3 data frames, when utilising the ADalphabet, or 1.5 data frames, for the application of the KGalphabets, were stored in a HTADcoder Tx. buffer which was of the same bit length to that of the ADcoder.

Table 7.5 shows that the prevention of Tx. buffer overflow/Rx. buffer underflow within a HTKGcoder involved the discarding/repeating of 3 data frames when ADalphabet was utilised and 4 data frames for the implementation which employed the KHalphabet. In this coder a data frame consisted of one amplitude-codeword and four tes-codewords. A tes-codeword within a HTKGcoder represented two epochs and therefore, the information contained in a HTKGcoders data frame was equal to that of a KGcoders data frame.

The HTKGcoder (of Table 7.5) employed 3.5 bits per epoch (for the application of the KGalphabet) or 4.5 bits per epoch (for the application of the ADalphabet) compared with 6 bits in the ADcoder. Therefore, for every data frame stored in the Tx. buffer of the KGcoder 1.3 data frames, when utilising the ADalphabet, or 1.7 data frames, for the application of the KHalphabet, were stored in a HTKGcoder Tx. buffer which was of the same bit length to that of the ADcoder.

The above, indicates that the HTADcoder and the HTKGcoder store a greater number of epochs than the ADcoder and KGcoder, respectively, for buffers of the same bit lengths. Therefore, the rate at which the Tx. buffer of the Hadamard Transform coders reached a condition of imminent overflow differed to that of the standard coders.

The perceptual effects introduced into the synthesised speech by Tx. buffer overflow/Rx. buffer underflow within the coders utilising Hadamard Transformations were therefore different to those of the ADcoder and KGcoder.

7.7 Informal Subjective Appraisal

In section 7.4 it was indicated that the TES and TES related real-time coders presented within this chapter, in the opinion of the author, are by no means the final versions and further research is still required. Therefore, it would have been unjust to have conducted quality and intelligibility assessments for each coder. However, having discussed the various coders it would have been equally unjust not too have commented upon the present quality and intelligibility achievable at set transmission rates and delay. The system parameters utilised in these assessments have been specified in Tables 7.1 to 7.5.

In the following comments, unless otherwise specified, the transmision (Tx.) delay was approximately 300ms.

With a Tx. rate of 15kb/s the speech output from the ADcoder sounded very granular with spasmodic clicking. In the female segments of speech this effect was more severe and some truncation of words was detected. The female spoken fricative of the word 'Yes' was very distorted, mainly by Tx. buffer overflow/Rx. buffer underflow, which resulted in the 'Yes' sounding like 'Yeah'. Other utterances such as the word 'Newcastle' sounded very gargled and the 'ch'

of 'Charles' was distorted such that it sounded very "squeaky". Overall, the speech was of moderate to fair quality, intelligible and speaker recognition was possible.

When the Tx. rate was reduced to 10kb/s the fricatives of the speech output were very distorted. Clearly, buffer underflow/overflow was playing a significant part in the introduction of distortions. The speech output was not considered to be of lower quality than that produced by the coder at 15kb/s but occasionally sounded slightly synthetic.

At a Tx. rate of 15kb/s the speech output from the KGcoder sounded very synthetic, almost musical, and was of low quality and intelligibility. Because the speech sounded so synthetic in the subjective listenings no differences in quality and intelligibility were readily perceived for a Tx. rate of 10kb/s.

For a Tx. rate of 15kb/s, the algorithm which employed ADalphabet and encoded amplitude information over groups of eight epochs was found to produced a higher level of background noise yet much cleaner speech to that of the ADcoder. Utterances sounded more 'crisp' with only the occasional 'clicking'. When the Tx. rate was set at 10kb/s the speech took on a more granular quality with 'clicking' occurring more frequently than at 15kb/s. The quality and intelligibility of this coder at either 10 or 15kb/s was considered to be superior to that of the ADcoder and vastly superior to that of the KGcoder for a Tx. delay of 300ms.

The speech output from the algorithm employing differential

amplitude signalling and the KHalphabet was of similar quality and intelligibility to that of the KGcoder at both 10 and 15kb/s.

For the application of the ADalphabet in this coder, at 15kb/s, the background and inter-word noise was found to be very prominent and distracting. The speech was "crackly" which was found to interfere/interupt the utterance. The speech was very distinct and of fair quality and intelligibily. For a Tx. rate of 10kb/s the background and inter-word noise was again very prominent and the 'crackly' sound had increased. This caused a subjective reduction in quality and intelligibility. The increase in speech distortion was attributed to an increase in the frequency with which the buffers were overflowing/underflowing.

There were no obvious differences in quality/intelligibility of the speech output by the HTADcoder with the KHalphabet and that output from the KGcoder at data Tx. rates of 10 or 15kb/s. The speech sounded synthetic and was of low intelligibility.

With the ADalphabet in the HTADcoder at 15kb/s the background noise was quite loud and the female utterances were not as distinct as the male. The speech output was of fair quality and intelligibility. When the Tx. rate was 10kb/s, the background noise sounded discontinuous, as though it was being interrupted. The speech was slightly 'garbled' during some utterances and was not as clear as that for 15kb/s data transmission. The male uttered 'Yes' was very indistinct and the female version was of similar quality. The speech output was considered to be of moderate to fair quality and intelligibility.

The HTADcoder with the ADalphabet was assessed for a data Tx. rate of 5kb/s. For Tx. delays of both 300 and 600ms the background noise was found to be very broken up and of a lower level to that for 10kb/s transmission. The speech output with a delay of 300ms was of low quality and intelligibility because it sounded very synthetic. For a delay of 600ms the speech sounded slightly less synthetic but was still of low quality and intelligibility.

The final group of algorithms were the HTKGcoders. Once again, with the KHalphabet implemented the quality was only marginally better than that of the KGcoder, for Tx. rates of 10 and 15kb/s.

With the ADalphabet implemented in the HTKGcoder the speech produced was slightly gargled and some words were not very clear because of the 'crackly' noise inparticular, the 'c' of the words 'centre' and 'code' were very distorted by this. For a Tx. rate of 10kb/s similar effects were heard and it was difficult to distinguish between the speech output at 10kb/s and 15kb/s. The speech quality and intelligibility at both data rates, was considered to be fair.

At a Tx. rate of 5kb/s and delays of 300 and 600ms the speech was 'crackly' with the occasional 'clicking'. This caused the speech to be subjectively perceived as of lower quality to that produced at 10 and 15kb/s. When the delay was 300ms the speech was considered to be slightly clearer than that output for a delay of 600ms.

7.8 Summary

In this chapter the real-time implementation of the Al-Doubooni and King and Gosling TES coders have been discussed. Also discussed were the TES related coders which have incorporated techniques for reducing the data required for signalling of epoch amplitude and duration information.

It was discovered that the subjective quality of the speech output from the Al-Doubooni TES coder was superior to that output by the King and Gosling TES coder at all Tx. data rates with a Tx. delay of 300ms. In all the real-time applications which employed the King and Holbeche coding alphabet the speech output had a very synthetic quality. The algorithm which synthesised the speech of, subjectively, the highest quality and intelligibility at 10 and 15kb/s was the Al-Doubooni coder with amplitude information signalled for groups of eight epochs. This technique yielded a 2:1 data reduction to that required within the original Al-Doubooni coder.

The coders which included Hadamard Transformations of the epoch duration sequence and the Al-Doubooni coding alphabet produced speech of moderate to fair quality and intelligibility at Tx. rates of 10 and 15kb/s and Tx. delay of 300ms. With the King and Holbeche coding alphabet implemented the speech output sounded synthetic and was of very poor quality and intelligibility.

Coder Parameters	ADcoder Alphabets	KGcoder Alphabets
No. of Amplitude Codewords	1	1
No. of bits in Amplitude codeword	9	8
No. of Tes codewords	1	8
No. of bits in Tes codeword	7	5
No. of data frames discarded	9	3
Filler word (No. of bits)	16	48
Extrema Sensetivity	- 44dB	
Transmission Delay (ms)	≈ 300	
Tx. Buffer Delay (ms)	≈ 100	
Rx. Buffer Delay (ms)	≈ 200	
Tx. Rates Investigated (kb/s)	{ 15 { { 10	

Table 7.1 : System parameters for the real-time implementations of the TES coders of section 7.2 and 7.3.

Coder Parameters	ADcoder Alphabets	KGcoder Alphabets
No. of Amplitude Codewords	1	1
No. of bits in Amplitude codeword	8	8
No. of Tes codewords	8	8
No. of bits in Tes codeword	7	5
No. of data frames discarded	3	4
Filler Word (No. of bits)	64	48
Extrema Sensetivity	- 44dB	
Transmission Delay (ms)	≈ 300	
Tx. Buffer Delay (ms)	≈ 100	
Rx. Buffer Delay (ms)	≈ 200	
Tx. Rates Investigated (kb/s)	{ 15 { { 10	

Table 7.2 : System parameters for the real-time implementations of the TES coders of section 7.5 with amplitude signalled for groups of 8 epochs.

Coder Parameters	ADcoder Alphabets	KGcoder Alphabets
No. of Amplitude Codewords	1	1
No. of bits in Amplitude codeword	2	2
No. of Tes codewords	1	1
No. of bits in Tes codeword	7	5
No. of data frames discarded	3	4
Filler Word (No. of bits)	16	16
Extrema Sensetivity	- 44dB	
Transmission Delay (ms)	≈ 300	
Tx. Buffer Delay (ms)	≈ 100	
Rx. Buffer Delay (ms)	≈ 200	
Tx. Rates Investigated (kb/s)	{ 15 { { 10	

Table 7.3 : System parameters for the real-time implementations of the TES coders of section 7.5 with differential amplitude signalling.

Coder Parameters	ADcoder Alphabets	KGcoder Alphabets
No. of Amplitude Codewords	2	2
No. of bits in Amplitude codeword	9	8
No. of Tes codewords	1	1
No. of bits in Tes codeword	7	5
No. of data frames discarded	4	6
Filler Word (No. of bits)	32	32
Extrema Sensetivity	- 44dB	
Transmission Delay (ms)	≈ 300	
Tx. Buffer Delay (ms)	≈ 100	
Rx. Buffer Delay (ms)	≈ 200	
Tx. Rates Investigated (kb/s)	{ 15 { { 10	

Table 7.4 : System parameters for the real-time implementations of the TES coders of section 7.6 with Hadamard Transformation of the coded shape descriptor.

Coder Parameters	ADcoder Alphabets	KGcoder Alphabets
No. of Amplitude Codewords	1	1
No. of bits in Amplitude codeword	8	8
No. of Tes codewords	4	4
No. of bits in Tes codeword	7	5
No. of data frames discarded	3	4
Filler Word (No. of bits)	32	32
Extrema Sensetivity	- 44dB	
Transmission delay (ms)	≈ 300/600	
Tx. Buffer Delay (ms)	≈ 100/200	
Rx. Buffer Delay (ms)	≈ 200/00	
Tx. Rates Investigated (kb/s)	{ 15 { { 10 { { 5	

Table 7.5 : System parameters for the real-time implementations of the TES coders of section 7.5 with Hadamard Transformation of the coded shape descriptor and group amplitude signalling.

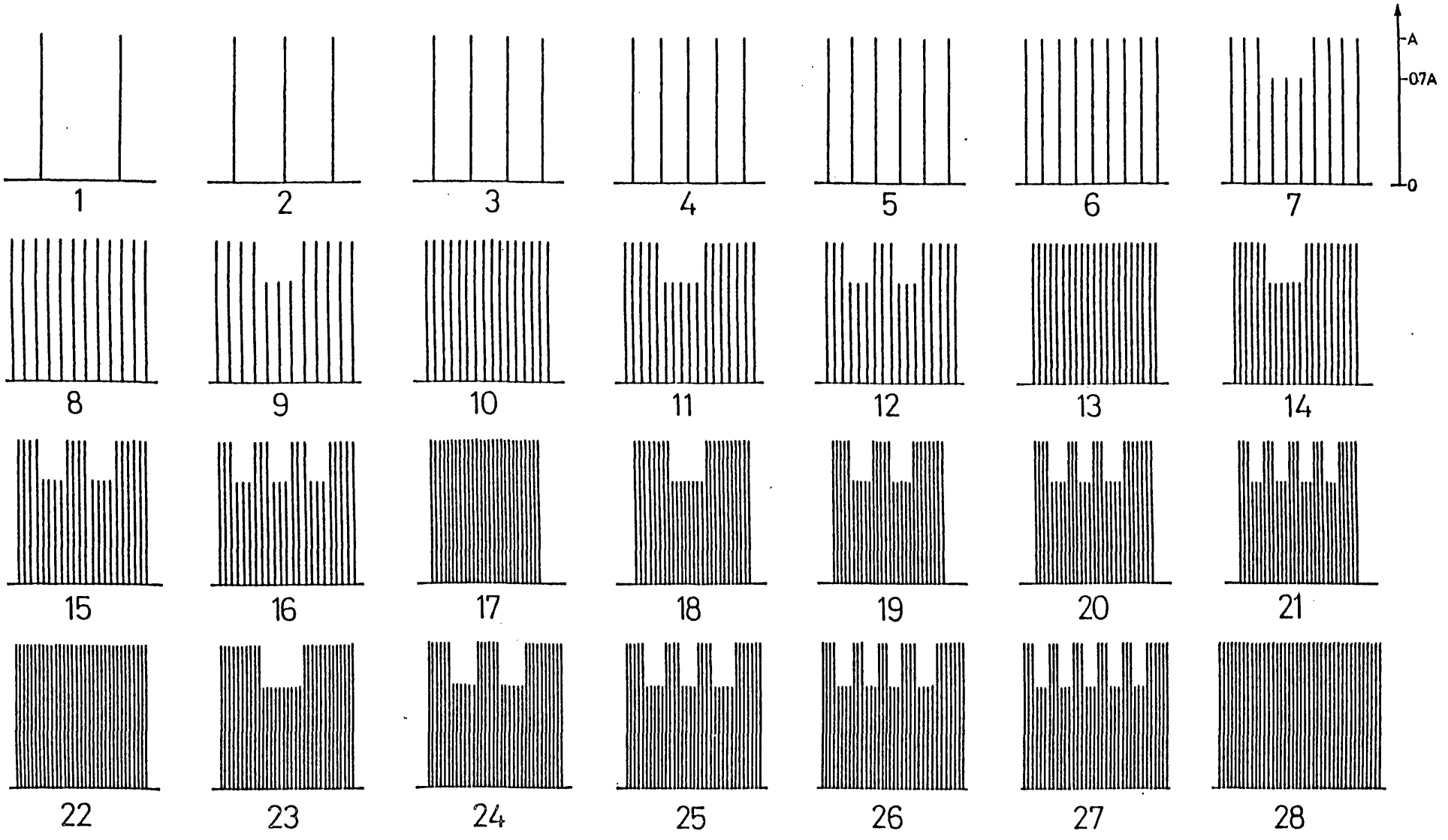


Figure 7.1 : 28 stylised waveshapes for the King and Holbeche coding alphabet [93].

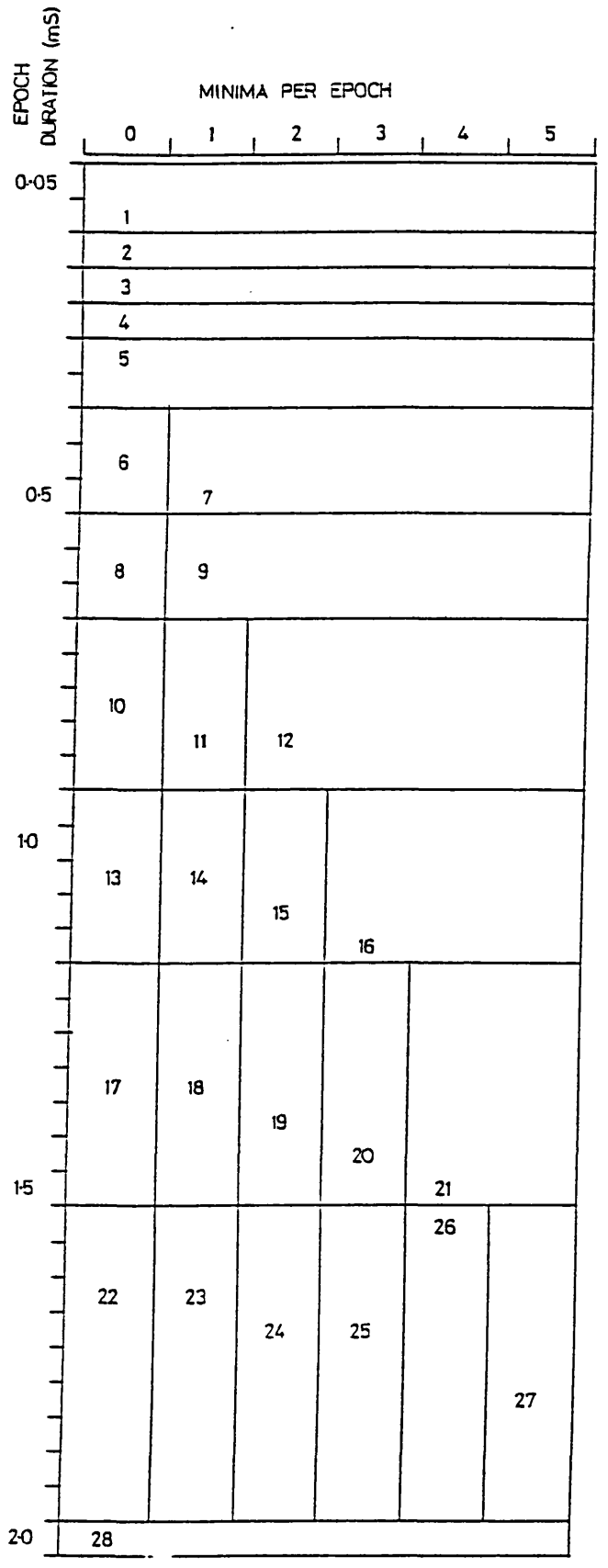


Figure 7.2 : King and Holbeche coding matrix for a real-time implementation.

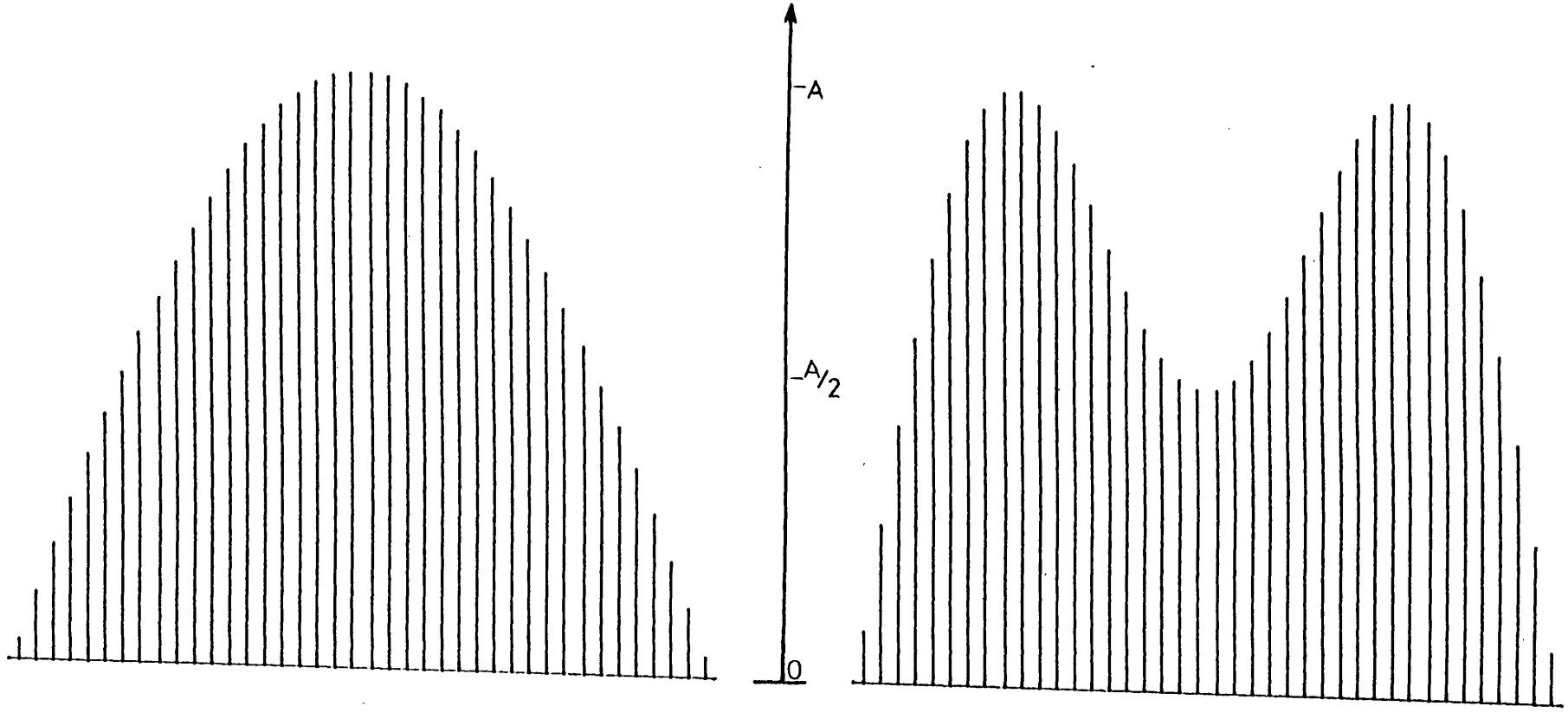


Figure 7.3 : Stylised waveshapes for Al-Doubooni coding alphabet [39].

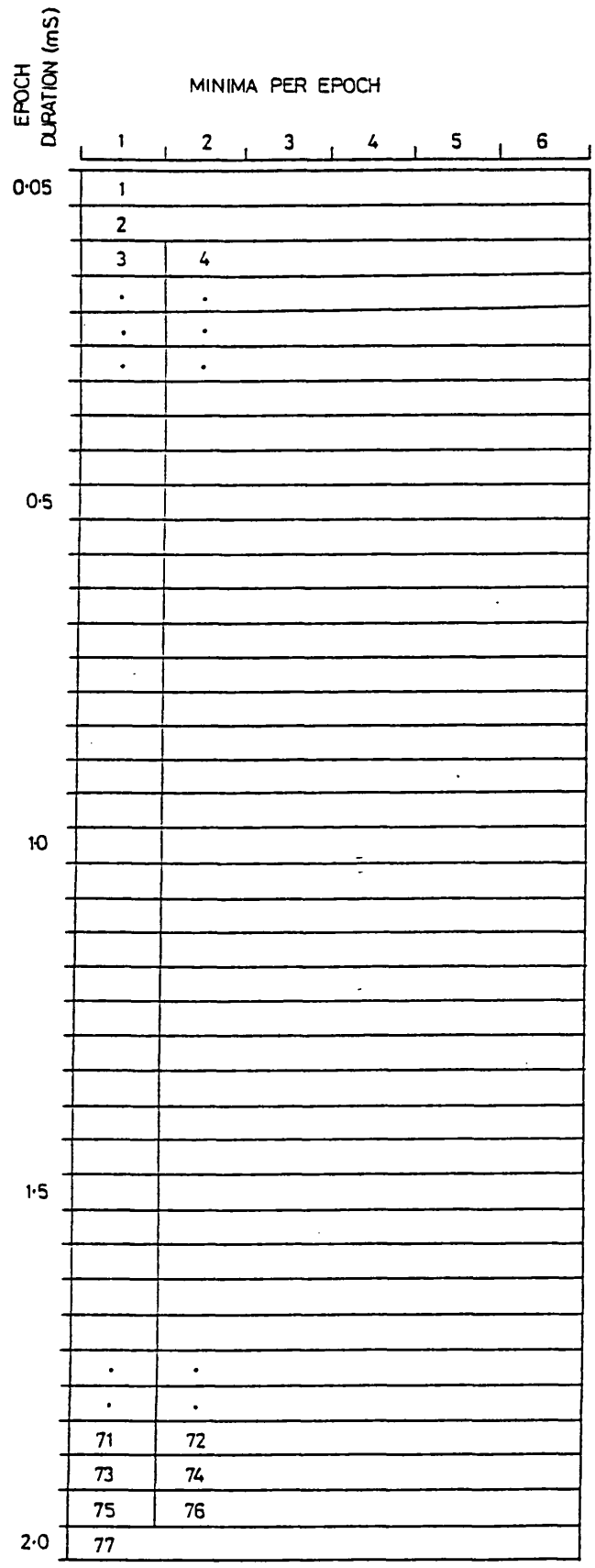
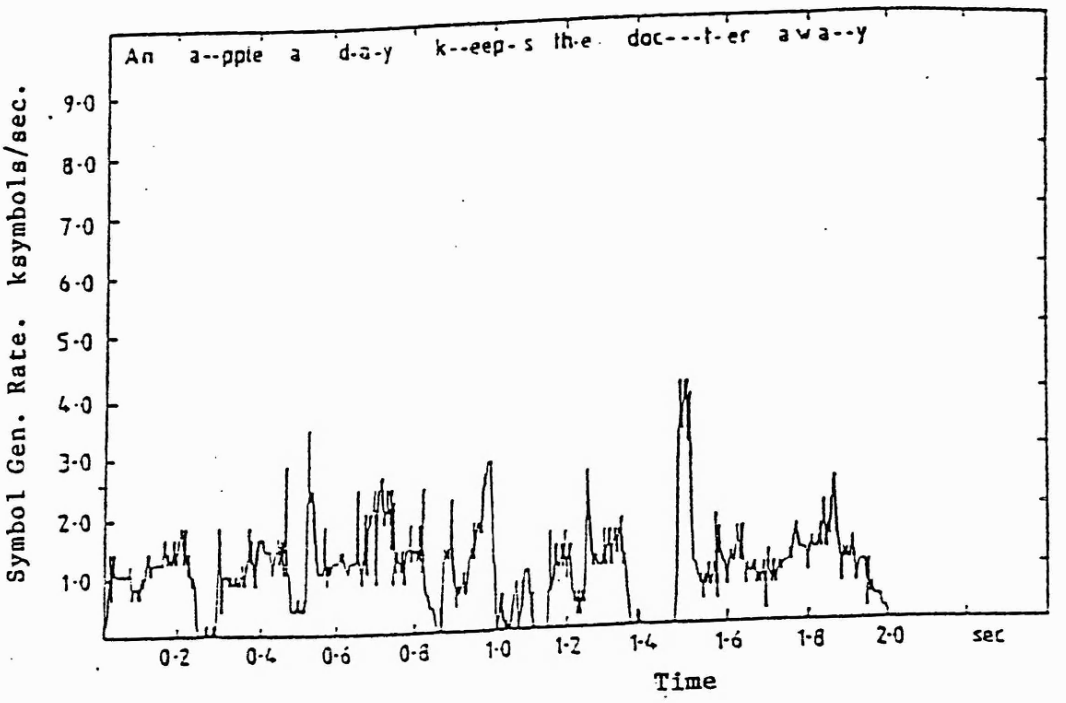


Figure 7.4 : Al-Doubooni coding matrix for a real-time implementation.



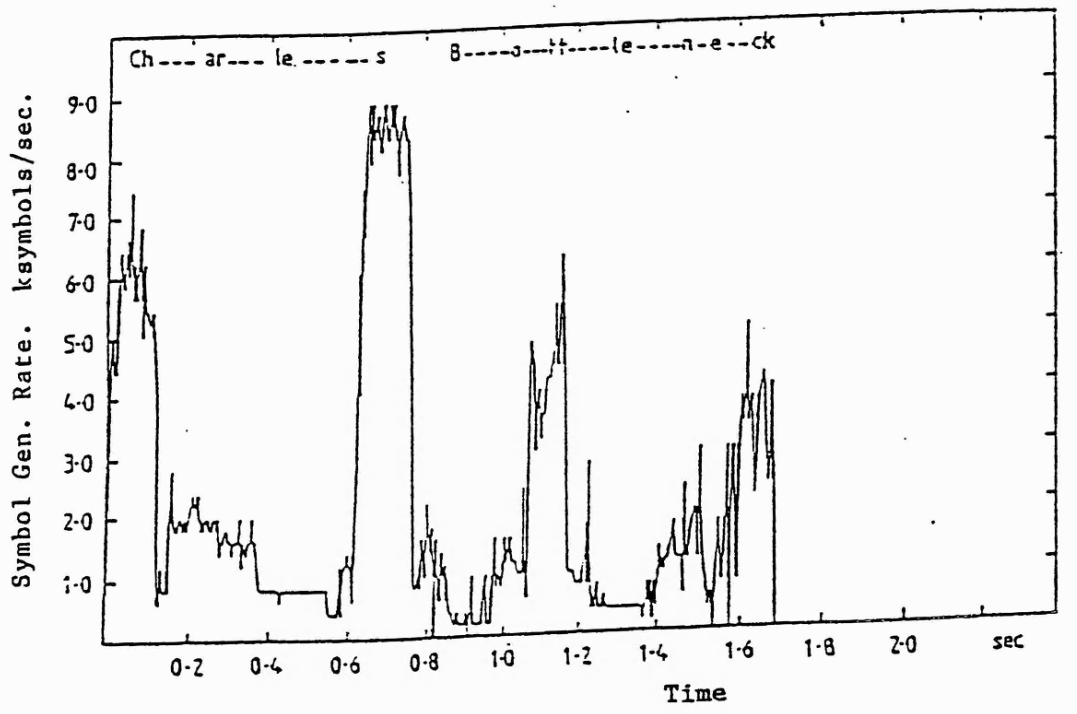


Figure 7.5(a),(b) : Symbol generation rates for the two utterances (After Seneviratne [97]).

Chapter 8

Conclusions and Recommendations For Further Research.

8.1 Conclusions.

This thesis has presented:

a) The development of a simplex digital voice channel which is a powerfull tool for the study of speech coders, and in particular TIME ENCODED SPEECH (TES), in real-time.

b) A study of techniques for the pre-processing of TES parameters for the enhancement of other data reduction techniques.

c) Investigations into techniques for achieving a reduction in the information requirements for the signalling of TES parameters.

and d) The real-time implementation of TES and TES related coders for various transmission bit rates and delays.

The real-time studies of chapter 3 have indicated a minimum of one bit per epoch as sufficient for conveying amplitude information for the synthesis of speech of acceptable quality. Diagnostic Rhyme Tests (DRT) yielded scores of the order of 88% intelligibility for algorithms which utilised one and two bits of amplitude information per epoch. The DRT scores, in both cases, were appreciably less than for the unprocessed control PCM samples. Direct Comparison Tests did not highlight any clear preference between the two coding techniques assessed in the DRT's.

These investigations were conducted such that only the amplitude parameter was distorted, all other TES parameters were undistorted and buffering was not incorporated. Therefore, it must be concluded that the true intelligibility from a TES system will be

dependant upon the amplitude encoding/decoding technique as well as a number of other features of the TES system.

The simulation investigations into median and moving average smoothing algorithms for the pre-processing of the epoch duration sequences demonstrated that only modest levels of smoothing may be applied to such sequencies before significant degradation in the speech quality occurs. Therefore, it must be concluded that the application of epoch sequence pre-processing which involves simple numerical smoothing, is of little value for it appears to degrade the quality of the synthesised speech in an unacceptable manner.

The simulation investigations into extremal coding have demonstrated that the restricted distance measure (r.d.m) extremal coding of the epoch durations (EXTR1) can yield data compression ratios of 1.36:1 (with an r.d.m of 3) and 1.33:1 (with an r.d.m of 7). The synthesised speech was informally judged to have retained a high degree of intelligibility but varied in quality. The characteristics of speakers, in particular the male utterances, were found to have altered.

When the peak magnitude sequences were pseudo-extremally coded (EXTR2) overall data compression ratios of 1.61:1 (with an r.d.m of 3) and 1.65:1 (with an r.d.m of 7). However, a feature of this form of coding was the "zeroing" of epoch peak amplitudes which manifested as silence in the synthesised speech. The overall quality was judged to be inferior to that produced from the output of EXTR1.

The investigations, which were conducted in simulation, into

Orthogonal Transformations of the epoch duration sequences yielded a maximum data reduction of 1.35:1 which was achieved employing Dominant Coefficient Retention in a 16 point Hadamard Transform. The resulting speech was intelligible but of moderate quality.

The speech synthesised from the Low-pass Sequency Filtered Hadamard coefficients was of high quality and intelligibility. A maximum data reduction of 1.33:1 was achieved for N = 4. However, analysis of this process demonstrated that an identical epoch duration sequence (to that output from the transform) is achieved if the average of pairs of adjacent epoch durations was calculated and the original epoch durations were replaced with the average value. This process yielded a 2:1 data reduction in the epoch duration sequence because only the average value need be transmitted. This process also reduced the 'transformation' delay from N epoch durations to two epoch durations, where N is the size of the Hadamard transform, ie. by a factor of N/2.

The real-time implementation of the Al-Doubooni and King and Gosling TES coder were presented in chapter 7. Also discussed were the TES related coders which have incorporated techniques for reducing the data required for signalling of epoch amplitude and duration information.

It was found for the system delays under investigation, viz. 600ms round trip delay, that the subjective quality of the speech output from the Al-Doubooni TES coder was superior to that output by the King and Gosling TES coder at all data Tx. rates. In all of the real-time applications which employed the King and Holbeche coding

alphabet the speech output had a very synthetic quality. The algorithm which synthesised the speech of, subjectively, the highest quality and intelligibility at 10 and 15kb/s was a modified Al-Doubooni coder with amplitude information signalled for groups of eight epochs. This technique yielded a 2:1 data reduction to that required within the original Al-Doubooni coder.

The coders which included Hadamard Transformations of the epoch duration sequence produced speech of moderate to fair quality and intelligibility.

From the research presented in this thesis it may be concluded that data reduction is possible in low complexity TES systems without producing speech of unacceptable quality. However, it is clear from the real-time investigations that further work is required in the development of TES algorithms which are less vulnerable to the mutilations inherent in real-time implementations before the quality of speech output in simulation exercises may be achieved in a real-time TES coder.

8.2 Recommendations for Further Research.

During the course of this thesis a number of aspects of Time Encoded Speech have been investigated. However, it is the authors opinion that further work is required in the following areas:

(1) The three main attributes of a TES system are: amplitude, epoch duration and shape. The amplitude information has been investigated in real-time while preventing the distortion of the other

attributes or the introduction of buffering and its associated problems. The remaining attributes also need to be studied independatly in real-time. The isolation of the results of all three investigations may aid the development of a superior coder to that of the present art.

(2) A powerful tool is now available to the researcher in the Time Encoding field, namely the simplex digital voice channel described in this thesis. Using this system, extensive studies of buffer management stratergies may be conducted to assess the effectiveness of, and distortion introduced by, such techniques. This system may also be utilised for the instantaneous comparisons of stylised waveform segments for speech synthesis, application of variable/run length coding, effects of system parameter variations or the development of coding alphabets.

(3) Median and moving average filters have been seen to be of little value when applied directly to the TES epoch duration sequences. However, the selective application of this technique via an "intelligent" algorithm may enhance its effectiveness and therefore be of interest for further work. Fricative segments of speech have the greatest symbol generation rate and the smoothing of such segments could enable more effective encoding.

(4) During unvoiced speech the spectral descriptors are slowly varying and only require updating over periods of the order of 20ms. The incorporation of a repeat codeword and an intelligent insertion routine may be used for the repetition of a 3ms segment of unvoiced speech. This could achieve data reductions of the order

of 6:1 and assist in the reduction of transmission buffer overflow. Such a method would also require a technique within the receiver for the prevention of the disturbing effects which result when a segment of speech is repeated several times.

APPENDICES

Appendix 1

Listing of Commented Mnemonic Code for Transmitter and Receiver Algorithms

```
;
;
;
;  Hardware Data and Peripheral Codes
;  ==================================
;
;  Analogue-to-Digital Converter(signals)
;  Board number = 00575
;  Operational mode:     16 single ended inputs
;                        12 bit resolution
;                        -10V to +10V input range
;                        2's complement representation
;
ADMULT: .EQ     #30             ;Multiplexer channel
ADCTRL: .EQ     #31             ;Control channel
ADSTAT: .EQ     #32             ;Status channel
ADIN:   .EQ     #33             ;Data channel
;
;  Parallel Interface Card
;  Board number = 00403
;
PAIN:   .EQ     #C0             ;Input channel A
PAOUT:  .EQ     #C1             ;Output channel A
PASTAT: .EQ     #C2             ;Status channel A
PACTRL: .EQ     #C3             ;Control channel A
PBIN:   .EQ     #C4             ;Input channel B
PBOUT:  .EQ     #C5             ;Output channel B
PBSTAT: .EQ     #C6             ;Status channel B
PBCTRL: .EQ     #C7             ;Control channel B
;
;  Hardware Multiplier Card
;  Board number = 00581
;
MX:     .EQ     #A0             ;Load X (no rounding)
MXR:    .EQ     #A1             ;Load X (rounding)
MY:     .EQ     #A2             ;Load Y
MXYMS:  .EQ     #A3             ;Dump product (ms word)
MXYLS:  .EQ     #A4             ;Dump product (ls word)
;
;  Serial Interface Card
;  Board number = 00401
;
SIN:    .EQ     #10             ;input data
SOUT:   .EQ     #11             ;output data
SRSTAT: .EQ     #12             ;receiver status
SRCTRL: .EQ     #13             ;receiver control
STSTAT: .EQ     #14             ;transmitter status
STCTRL: .EQ     #15             ;transmitter control
SBSTAT: .EQ     #16             ;baud rate status
SBCTRL: .EQ     #17             ;baud rate control
;
```

```
;
;
;       Transmitter algorithm
;       =====================
;
;
ADVEA:  .DS     #0,1            ;volatile environment - ISRAD
ADVEE:  .DS     #0,1
ADVEX:  .DS     #0,1
;
TXVEA:  .DS     #0,1            ;volatile environment - ISRTX
TXVEE:  .DS     #0,1
TXVEX:  .DS     #0,1
;
L1:     .DC     #0,#0100        ;*** AD buffer length ***
B1:     .DC     #0,#0B00        ;** AD buffer start address *
R1:     .DS     #0,1            ;AD buffer read pointer
W1:     .DS     #0,1            ;AD buffer write pointer
;
L2:     .DC     #0,#0040        ;*** TX buffer length ***
B2:     .DC     #0,#0C00        ;** TX buffer start address *
R2:     .DS     #0,1            ;TX buffer read pointer
W2:     .DS     #0,1            ;TX buffer write pointer
;
LD:     .DC     #0,#0002        ;Number of FRAMES to be discar
BD:     .DC     #0,#00E0        ;Delay buffer start address
PD:     .DS     #0,1            ; READ/WRITE pointer
;
ADCHAN: .DC     #0,#0000        ;ADC multiplexer address
ADTRIG: .DC     #0,#1C80        ;ADC control word
ADOFF:  .DC     #0,#0000        ;** ADC offset **
;
PATRIG: .DC     #0,#00C0        ;parallel interface A control
PBTRIG: .DC     #0,#0004        ;parallel interface B control
;
STMASK: .DC     #0,#673B        ;serial interface status
;
S0:     .DS     #0,1            ;current sample
S1:     .DS     #0,1            ;previous sample
SS1:    .DS     #0,1            ;last non-zero sample
SMIN:   .DC     #0,#0000        ;zero-crossing sensitivity
DS0:    .DS     #0,1            ;current S0-S1
DS1:    .DS     #0,1            ;previous S0-S1
DSS1:   .DS     #0,1            ;last non-zero difference
DSMIN:  .DC     #0,#0000        ;extremum sensitivity
;
IS:     .DS     #0,1            ;sample counter
MS:     .DS     #0,1            ;maximum number of samples
IC:     .DS     #0,1            ;extremum counter
MC:     .DS     #0,1            ;maximum number of extrema
;
A:      .DS     #0,1            ;amplitude
AMAX:   .DC     #0,#07FF        ;maximum amplitude
;
ACODE:  .DC     #0,#0100        ;** amplitude look up encoder
TCODE:  .DC     #0,#0900        ;** tes look up encoder **
```

```
;
IAW:     .DS     #0,1            ;amplitude word counter
NAW:     .DC     #0,#0001        ;*** number amplitude words ***
;
ITW:     .DS     #0,1    ;       ;tes word counter
NTW:     .DC     #0,#0008        ;*** number of tes words ***
;
NSW      .DC     #0,#0004        ;no. 12 bit words to form filler
ISYNC    .DS     #0,1            ;no. frames since synchronisation
NSYNC    .DC     #0,#0100        ;max. value of ISYNC
FWORD    .DS     #0,1            ;filler word status
;
IB:      .DS     #0,1            ;bit counter for data
BITS:    .DS     #0,1            ;input data for transmission
;
WARN:    .DC     #0,#000D        ;ASCII codes for error message
         .DC     #0,#000A
         .DC     #0,#0045
         .DC     #0,#0072
         .DC     #0,#0072
         .DC     #0,#006F
         .DC     #0,#0072
         .DC     #0,#0020
         .DC     #0,#004E
         .DC     #0,#0075
         .DC     #0,#006D
         .DC     #0,#0062
         .DC     #0,#0065
         .DC     #0,#0072
         .DC     #0,#0020
         .DC     #0,#0000
;
FRAME:   .DS     #0,1            ;transmitter frame
```

```
;
;
;
;  Initialisation Routine
;  ======================
;
START:  IOF                         ;disable interrupts
        CLRA                        ;unmask all levels
        OTA     #01
        PAGE    #0,#0
;
        CLRA                        ;zero variables
        STA     R1
        STA     R2
        STA     S0
        STA     S1
        STA     SS1
        STA     DS0
        STA     DS1
        STA     DSS1
        STA     A
        STA     IAW
        STA     ITW
        STA     IB
        STA     PD
        STA     FWORD
;
        LDA     TCODE
        RAX
        LDEW
        STE     MS
        INCA
        RAX
        LDEW
        STE     MC
;
        LCA     #01                 ;initialise counters
        STA     IS
        STA     IC
;
        CLRA                        ;zero AD buffer
        STA     W1
        LCE     #00
AGAIN1: LDA     W1
        ADD     B1
        RAX
        STEW
        LDA     W1
        INCA
        SUB     L1
        SKZP
        ADD     L1
        STA     W1
        SKZ
        JMPC    AGAIN1
;
```

```
        CLRA                    ;zero TX buffer
        STA     W2
        LCE     #00
AGAIN2: LDA     W2
        ADD     B2
        RAX
        STEW
        LDA     W2
        INCA
        SUB     L2
        SKZF
        ADD     L2
        STA     W2
        SKZ
        JMPC    AGAIN2
;
        CLRA                    ; Zero discard buffer
        STA     FD
        LCE     #0
AGAIN3: LDA     FD
        ADD     BD
        RAX
        STEW
;
        LDA     FD              ;Increment pointer
        INCA
        SUB     LD
        SKZF
        ADD     LD
        STA     FD
        SKZ
        JMPC    AGAIN3
;
        LDA     ADCHAN          ;set up ADC
        OTA     ADMULT
        LDA     ADTRIG
        OTA     ADCTRL
;
        LDA     PATRIG          ;set up parallel channel A
        OTA     PACTRL
;
        ION                     ;enable interrupts
        JMPC    LOOP            ;start processing
```

```
;
;
;
;  Error Message Generation
;  ========================
;
ERROR:  IOF
        CLRA
        RAX
;
ER10:   LDAX    WARN
        SKNZ
        JMPC    ER20
;
        OTA     SOUT
ER11:   INA     STSTAT
        ERA     STMASK
        SKZ
        JMPC    ER11
;
        NOPT
        NOP
        JMPC    ER10
;
ER20:   LCA     #30
        ADDE
        OTA     SOUT
ER21:   INA     STSTAT
        ERA     STMASK
        SKZ
        JMPC    ER21
;
ER30:   LCA     #0D
        OTA     SOUT
ER31:   INA     STSTAT
        ERA     STMASK
        SKZ
        JMPC    ER31
;
ER40:   LCA     #0A
        OTA     SOUT
ER41:   INA     STSTAT
        ERA     STMASK
        SKZ
        JMPC    ER41
;
ER50:   JMPC    ER50
```

```
;
;
;
;  Interrupt Vector Table
;  ======================
;
        .LOC    #00F0
;
        IONR    #F0
        IONR    #F1
        IONR    #F2
        JMPC    ISRAD           ;ISR:Analogue-to-Digital
        IONR    #F4
        JMPC    ISRTX           ;ISR:Transmission
        IONR    #F6
        IONR    #F7
        IONR    #F8
        IONR    #F9
        IONR    #FA
        IONR    #FB
        IONR    #FC
        IONR    #FD
        IONR    #FE
        IONR    #FF
```

```
;
;
;
;  Analogue-to-Digital Interrupt Service Routine
;  =============================================
;
;
ISRAD:  STA     ADVEA           ;save volatile environment
        STE     ADVEE
        STX     ADVEX
;
AD11:   INA     ADIN            ;input sample
        SUB     ADOFF           ;adjust for dc offset
        RAE
;
AD12:   LDA     W1              ;store sample
        ADD     B1
        RAX
        STEW
;
AD13:   LDA     W1              ;increment pointer
        INCA
        SUB     L1
        SKZP
        ADD     L1
        STA     W1
;
AD14:   LDA     W1              ;test for imminent overflow
        SUB     R1
        LCE     #01
        SKNZ
        JMPC    ERROR
;
AD15:   LDX     ADVEX           ;restore volatile environment
        LDE     ADVEE
        LDA     ADVEA
        IONR    #F3
```

```
;
;
;
;  Transmitter Interrupt Service Routine
;  =====================================
;
ISRTX:  STA     TXVEA           ;save volatile environment
        STE     TXVEE
        STX     TXVEX
        INA     FASTAT          ;clear interrupt
        CLRA                    ;mask interrupts at this level
        BSET    #2
        OTA     #01
        ION                     ;enable unmasked interrupts
;
TX2:    LDA     IB              ;test for new input
        SKZ
        JMPC    TX18            ;jump if more bits remain
;
        LDA     W2              ;test for empty tx buffer
        SUB     R2
        SKNZ
        JMPC    TX10            ;insert filler word
;
        CLRA
        STA     FWORD
;
TX4:    LDA     R2              ;read next word
        ADD     B2
        RAX
        LDEW
;
TX6:    LDA     R2              ;increment pointer
        INCA
        SUB     L2
        SKZF
        ADD     L2
        STA     R2
;
TX8:    CLRA                    ;seperate fields
        SLE
        SLE
        SLE
        SLE
        STA     IB
        STE     BITS
        JMPC    TX18
;
TX10:   RPA
        ADDC    #04
        ADD     FWORD
        JMA
        JMPC    TX12
        JMPC    TX14
        JMPC    TX16
;
```

```
TX12:   LCA     #01             ;assume next filler word
        STA     FWORD
        LCA     #10             ;bits 0 - 15 of filler-word
        STA     IB
        LCA     -1              ; 1111 1111 1111 1111
        STA     BITS
        JMPC    TX18
;
TX14:   LCA     #02
        STA     FWORD
        LCA     #10             ;bits 16 - 31 of filler word
        STA     IB
        LCA     -1              ; 1111 1111 1111 1111
        STA     BITS
        JMPC    TX18
;
TX16:   CLRA
        STA     FWORD
        LCA     #10             ;bits 32 - 47 of filler word
        STA     IB
        LCA     -2              ; 1111 1111 1111 1110
        STA     BITS
;
TX18:   LDA     BITS
        OTA     PAOUT           ;output data on channel A
        SLE
        STA     BITS
        LDA     IB              ;decrement input bit count
        DECA
        STA     IB
;
TX19:   IOF                     ;disable interrupts
        CLRA                    ;unmask interrupts at this level
        OTA     #01
;
TX20:   LDX     TXVEX           ;restore volatile environment
        LDE     TXVEE
        LDA     TXVEA
        IONR    #F5
```

```
;
;
;
;  Main Analysis Loop
;  ==================
;
LOOP:   LDA     W1              ;check for empty sample buffer
        SUB     R1
        SKNZ
        JMPC    LOOP
;
AN11:   LDA     R1              ;fetch next sample
        ADD     B1
        RAX
        LDAW
        STA     S0
;
        LDA     R1              ;increment read pointer
        INCA
        SUB     L1
        SKZP
        ADD     L1
        STA     R1
;
AN12:   LDA     S0              ;update extemum count
        SUB     S1
        STA     DS0
;
        CLRA
        SUB     DS0
        SKZP
        LDA     DS0
        SUB     DSMIN           ; ABS( DS0 ).GE.DSMIN
        SKZP
        JMPC    AN12A           ; .FALSE.
        LDA     DS0             ; .TRUE.
        LDE     DSS1
        STA     DSS1
        ERAE                    ;test for change in direction
        BSKO    #F
        JMPC    AN12A
        LDA     IC              ;increment extreme count
        INCA
        STA     IC
        LCE     #07
        SKZP
        JMPC    ERROR
;
AN12A:  CLRA
        SUB     S0
        SKZP
        LDA     S0
        SUB     SMIN            ; ABS( S0 ).GE.SMIN
        SKZP
        JMPC    AN13
;
```

```
        LDA     S0              ;change of sign
        LDE     SS1
        STA     SS1
        ERAE
        BSKZ    #F
        JMPC    AN20
;
AN13:   LDA     IS              ;increment sample counter
        INCA
        STA     IS
        LCE     #08
        SKZP
        JMPC    ERROR
;
AN15:   CLRA                    ;update amplitude measure
        SUB     S0
        SKZP
        LDA     S0
        RAE
        SUB     A
        SKN
        STE     A
;
AN16:   LDA     S0              ;update working variables
        STA     S1
        LDA     DS0
        STA     DS1
;
        JMPC    LOOP
```

```
;
;
;
;  End of Epoch Routine
;  ====================
;
AN20:   LDA     IC              ;truncate component count
        SRZ
        INCA
        STA     IC
;
        LDA     IC              ; IC > max. value (MC) ?
        LDE     MC
        SUBE
        SKN
        STE     IC              ;.TRUE. set IC = MC
;
        LDA     IC              ; IC < 1 ?
        LCE     #01
        SUBE
        SKZP
        STE     IC              ;.TRUE. set IC = 1
;
AN23:   LDA     IS              ;truncate sample count
        LDE     MS
        SUBE
        SKN
        STE     IS
;
        LDA     IS
        LCE     #01
        SUBE
        SKZP
        STE     IS
;
        LDA     TCODE           ;encode extrema/sample count
        ADDC    #2
        ADD     IC
        DECA
        RAX
        LDAW
        ADD     TCODE
        ADD     IS
        DECA
        RAX
        LDEW
;
AN25:   CLRA                    ;reset epoch parameters
        STA     IS
        STA     IC
;
AN26:   LDX     ITW
        STEX    FRAME
;
AN27:   LDA     ITW             ;increment epoch count
        INCA
```

```
SUB     NTW
SKZP
ADD     NTW
STA     ITW
SKZ                     ;test for end of frame
JMPC    AN13            ;False - process next sample
JMPC    AN30            ;True  - process frame
```

```
;
;
;
;  Standard End of Frame Routine
;  =============================
;
;
AN30:   LDE     AMAX            ;encode amplitude
        LDA     A
        SUBE
        SKZP
        LDE     A
        LDA     ACODE
        ADDC    #1
        ADDE
        RAX
        LDEW
        CLRA                    ;reset amp. measure
        STA     A
;
AN33:   LDA     W2              ;tx buffer full ?
        SUB     R2
        SKZP
        ADD     L2
        ADD     NAW
        ADD     NTW
        SUB     L2
        SKZP
        JMPC    AN34            ;False
;
        LDA     PD              ;True - retreive delayed pointer
        ADD     BD
        RAX
        LDAW
        STA     W2
;
AN34:   LDA     W2              ;write amplitude code-word
        ADD     B2
        RAX
        STEW
;
        LDA     PD              ;Store pointer in discard buffer
        ADD     BD
        RAX
        LDA     W2
        STAW
;
        LDA     PD              ;increment delay pointer
        INCA
        SUB     LD
        SKZP
        ADD     LD
        STA     PD
;
AN35:   LDA     W2              ;increment write pointer
        INCA
```

```
        SUB     L2
        SKZP
        ADD     L2
        STA     W2
;
AN36:   LDX     ITW             ;transcribe frame
        LDEX    FRAME
        LDA     W2
        ADD     B2
        RAX
        STEW
;
AN37:   LDA     W2              ;increment write pointer
        INCA
        SUB     L2
        SKZP
        ADD     L2
        STA     W2
;
AN38:   LDA     ITW             ;increment frame pointer
        INCA
        SUB     NTW
        SKZP
        ADD     NTW
        STA     ITW
        SKZ
        JMPC    AN36
;
AN40:   LDA     ISYNC           ;increment frame count
        INCA
        SUB     NSYNC
        SKZP
        ADD     NSYNC
        STA     ISYNC
        SKZ
        JMPC    AN13
;
        LCA     -1              ;load bits 0 - 11 of fillerword
        BCLR    #D
        BCLR    #C
        RAE
;
        LDA     W2              ;store fillerword in tx. buffer
        ADD     B2
        RAX
        STEW
;
        LDA     W2              ;increment write pointer
        INCA
        SUB     L2
        SKZP
        ADD     L2
        STA     W2
;
        LCA     -1              ;load bits 12 - 23 of fillerword
        BCLR    #D
```

```
        BCLR    #C
        RAE
;
        LDA     W2              ;store fillerword in tx. buffer
        ADD     B2
        RAX
        STEW
;
        LDA     W2              ;increment write pointer
        INCA
        SUB     L2
        SKZP
        ADD     L2
        STA     W2
;
        LCA     -1              ;load bits 24 - 35 of fillerword
        BCLR    #D
        BCLR    #C
        RAE
;
        LDA     W2              ;store fillerword in tx. buffer
        ADD     B2
        RAX
        STEW
;
        LDA     W2              ;increment write pointer
        INCA
        SUB     L2
        SKZP
        ADD     L2
        STA     W2
;
        LCA     -1              ;load bit 36 - 47 of fillerword
        BCLR    #D
        BCLR    #C
        BCLR    #O
        RAE
;
        LDA     W2              ;store fillerword in tx. buffer
        ADD     B2
        RAX
        STEW
;
        LDA     W2              ;increment write pointer
        INCA
        SUB     L2
        SKZP
        ADD     L2
        STA     W2
;
        JMPC    AN13
        .END
```

```
;
;
;
;       Receiver alosorithm
;       ===================
;
;
RXVEA:  .DS     #0,1            ;volatile environment - ISRRX
RXVEE:  .DS     #0,1
RXVEX:  .DS     #0,1
;
DAVEA:  .DS     #0,1            ;volatile environment - ISRDA
DAVEE:  .DS     #0,1
DAVEX:  .DS     #0,1
;
ADCHAN: .DC     #0,#0000        ;ADC multiplexer channel
ADTRIG: .DC     #0,#1C80        ;ADC control word
;
PBTRIG: .DC     #0,#0008        ;parallel interface A control word
;
STMASK: .DC     #0,#673B        ;serial interface status
;
L3:     .DC     #0,#0100        ;*** RX buffer length ***
B3:     .DC     #0,#0700        ;** RX buffer start address **
R3:     .DS     #0,1            ;RX buffer read pointer
W3:     .DS     #0,1            ;RX buffer write pointer
;
LR:     .DC     #0,#0002        ;Number of FRAMES repeated
BR:     .DC     #0,#00E0        ;Repeat buffer start address
PR:     .DS     #0,1            ;READ/WRITE pointer
;
L4:     .DC     #0,#0100        ;*** DA buffer length ***
B4:     .DC     #0,#0100        ;** DA buffer start address **
R4:     .DS     #0,1            ;DA buffer read pointer
W4:     .DS     #0,1            ;DA buffer write pointer
;
IAW:    .DS     #0,1            ;amplitude word counter
IAW1:   .DS     #0,1            ;amplitude word counter (ISRRX)
NAW:    .DC     #0,#0001        ;*** number of amplitude words ***
;
ITW:    .DS     #0,1            ;tes word counter
ITW1:   .DS     #0,1            ;tes word counter (ISRRX)
NTW:    .DC     #0,#0008        ;*** number of TES words ***
;
IBA:    .DS     #0,1            ;bit counter for amplitude word
MBA:    .DS     #0,1            ;maximum # amplitude bits
IBT:    .DS     #0,1            ;bit counter for tes word
MBT:    .DS     #0,1            ;maximum # for test bits
;
IA:     .DS     #0,1            ;amplitude dictionary index
IA1:    .DS     #0,1            ;amplitude dictionary index (ISRRX
NA:     .DS     #0,1            ;size of amplitude dictionary
IT:     .DS     #0,1            ;tes dictionary index
IT1:    .DS     #0,1            ;tes dictionary index
NT:     .DS     #0,1            ;size of tes dictionary
;
```

```
ACODE:   .DC     #0,#0200
TCODE:   .DC     #0,#0400
;
BITS:    .DS     #0,1            ;output from bit handler
LAST0:   .DS     #0,1            ;sequence buffer
LAST1:   .DS     #0,1            ;sequence buffer
LAST2:   .DS     #0,1            ;sequence buffer
MASK0:   .DC     #0,#FFFE        ;*** synchronisation mask ***
MASK1:   .DC     #0,#FFFF        ;*** synchronisation mask ***
MASK2:   .DC     #0,#FFFF        ;*** synchronisation mask ***
FWORD0:  .DC     #0,#FFFE        ;*** filler word ***
FWORD1:  .DC     #0,#FFFF        ;*** filler word ***
FWORD2:  .DC     #0,#FFFF        ;*** filler word ***
NBF:     .DC     #0,#0030        ;*** #bits in filler word ***
IBF:     .DS     #0,1            ;status of 1 word buffer
RXWD2:   .DS     #0,1            ;input to decoders
;
FLAG:    .DS     #0,1            ;status flag
DICT:    .EQ     #E              ;#E set if decoding amplitude
SYNC:    .EQ     #F              ;#F set if synchronised
;
IS:      .DS     #0,1            ;sample counter
NS:      .DS     #0,1            ;number of samples in epoch
MS:      .DC     #0,#0040        ;*** maximum #samples in epoch **
A:       .DS     #0,1            ;amplitude
DM:      .DS     #0,1            ;address of synthetic sample
;
WARN:    .DC     #0,#000D        ;ASCII codes for error message
         .DC     #0,#000A
         .DC     #0,#0045
         .DC     #0,#0072
         .DC     #0,#0072
         .DC     #0,#006F
         .DC     #0,#0072
         .DC     #0,#0020
         .DC     #0,#004E
         .DC     #0,#0075
         .DC     #0,#006D
         .DC     #0,#0062
         .DC     #0,#0065
         .DC     #0,#0072
         .DC     #0,#0020
         .DC     #0,#0000
;
```

```
;
;
;  Initialisation
;  ==============
;
;
START:  IOF
        CLRA
        OTA     #01
        PAGE    #0,#0
;
        CLRA                        ;initialise working variables
        STA     R3
        STA     R4
;
        STA     IS
        STA     PR
;
        STA     IAW
        STA     ITW1
        STA     ITW
;
        STA     BITS
        STA     LAST0
        STA     LAST1
        STA     FLAG
;
        LDA     NBF
        STA     IBF
;
        CLRA                        ;initialise Rx. buffer
        STA     W3
        LCE     #00
AGAIN1: LDA     W3
        ADD     B3
        RAX
        STEW
        LDA     W3
        INCA
        SUB     L3
        SKZF
        ADD     L3
        STA     W3
        SKZ
        JMPC    AGAIN1
;
        CLRA                        ;initialise o/p sample buffer
        STA     W4
        LCE     #00
AGAIN2: LDA     W4
        ADD     B4
        RAX
        STEW
        LDA     W4
        INCA
        SUB     L4
```

```
        SKZP
        ADD     L4
        STA     W4
        SKZ
        JMPC    AGAIN2
        LCA     #01
        STA     R4
;
NEXT0:  LDA     W3              ;initialise Rx. buffer with
        ADD     B3              ;dummy signal
        RAX
        LCA     #10             ;Store amplitude in Rx buffer
        BSET    #E
        STAW
;
        LDA     PR              ; Store W3 in delay buffer
        ADD     BR
        RAX
        LDA     W3
        STAW
;
        LDA     W3
        INCA
        SUB     L3
        SKZP
        ADD     L3
        STA     W3
;
        LCE     #08
NEXT2:  LDA     W3
        ADD     B3
        RAX                     ; Store epochs associated
        STEW                    ; with previously stored amp.
;
        LDA     W3              ;Increment pointer
        INCA
        SUB     L3
        SKZP
        ADD     L3
        STA     W3
;
        LDA     ITW             ;Test for ` end of frame `
        INCA
        SUB     NTW
        SKZP
        ADD     NTW
        STA     ITW
        SKZ
        JMPC    NEXT2
;
        LDA     PR              ;Increment delay pointer
        INCA
        SUB     LR
        SKZP
        ADD     LR
        STA     PR
```

```
        SKZ
        JMPC    NEXTO
;
        LCA     #FF
        STA     A
;
        LDA     ACODE           ;set size of Amp. dict.
        RAX
        LDEW
        STE     NA
        INCA
        RAX
        LDEW
        STE     MBA             ;set max #Amp bits
;
        LDA     TCODE           ;set size of Tes dict.
        RAX
        LDEW
        STE     NT
        INCA
        RAX
        LDEW
        STE     MBT             ;set max. #Tes bits
;
        CLRA
        STA     IBA
        STA     IBT
;
        LDA     ADCHAN
        OTA     ADMULT
        LDA     ADTRIG
        OTA     ADCTRL
;
        LDA     PBTRIG
        OTA     PBCTRL
;
        ION
        JMPC    LOOP
```

```
;
;
;
;  Interrupt Vector Table
;  ======================
;
;
        .LOC    #00F0
;
        IONR    #F0
        IONR    #F1
        IONR    #F2
        JMPC    ISRDA           ;ISR:Digital-to-Analogue
        IONR    #F4
        IONR    #F5
        IONR    #F6
        JMPC    ISRRX           ;ISR:Receiver
        IONR    #F8
        IONR    #F9
        IONR    #FA
        IONR    #FB
        IONR    #FC
        IONR    #FD
        IONR    #FE
        IONR    #FF
```

```
;
;
;
;  Digital-to-Analogue Interrupt Service Routine
;  =============================================
;
ISRDA:  STA     DAVEA           ;save volatile environment
        STE     DAVEE
        STX     DAVEX
;
DA11:   LDA     W4              ;test for empty sample buffer
        SUB     R4
        SKZP
        ADD     L4
        LCE     #06
        SKNZ
        JMPC    ERROR
;
DA12:   LDA     R4              ;fetch and output sample
        ADD     B4
        RAX
        LDAW
        OTA     DA1OUT
;
DA13:   LDA     R4              ;increment read pointer
        INCA
        SUB     L4
        SKZP
        ADD     L4
        STA     R4
;
DA14:   INA     ADIN            ;clear ADC interrupt
;
DA15:   LDX     DAVEX           ;resore volatile environment
        LDE     DAVEE
        LDA     DAVEA
        IONR    #F3
```

```
;
;
;
;  Receiver Interrupt Service Routine
;  ==================================
;
;  Synchronisation
;
ISRRX:  STA     RXVEA           ;save volatile environment
        STE     RXVEE
        STX     RXVEX
        INA     PBSTAT          ;clear interrupt
        CLRA                    ;mask interrupts at this level
        BSET    #02
        OTA     #01
        ION                     ;enable interrupts
;
RX11:   LDA     LAST2           ;update delay bits
        LDE     LAST1
        SLE
        STA     LAST2
        LDA     LAST0
        LDE     LAST1
        SLE
        STA     LAST0
;
        INA     PBIN            ;input word from channel B
        RAE
        LDA     LAST1
        SLE
        STA     LAST1
;
RX21:   LDA     IBF             ;update state of sequence buffer
        DECA
        SKN
        STA     IBF
;
RX22:   LDA     IBF             ;test for full sequence buffer
        SKZ
        JMPC    RX14            ;jump if delay not established
;
RX23:   LDA     LAST2
        ERA     FWORD2          ;test against filler sequence
        AND     MASK2
        SKZ
        JMPC    RX25
        LDA     LAST1
        ERA     FWORD1
        AND     MASK1
        SKNZ
        JMPC    RX26
        LDA     LAST0
        ERA     FWORD0
        AND     MASK0
        SKNZ
        JMPC    RX26
```

```
;
RX24:   LDA     MBA             ;set amplitude count
        STA     IBA
        CLRA
        STA     IBT
        STA     BITS
        STA     ITW1
        STA     IAW1
;
        LDA     NBF             ;clear  delay
        STA     IBF
;
        LDA     FLAG            ;synchronisation
        BSET    SYNC
        BSET    DICT
        STA     FLAG
        JMPC    RX14
;
RX25:   LDA     FLAG
        BSKO    SYNC
        JMPC    RX14
        BSKZ    DICT
        JMPC    RX30A
        JMPC    RX30T
;
RX14:   IOF                     ;disable interrupts
        CLRA                    ;unmask interrupts at this level
        OTA     #01
        INA     PBSTAT          ;check that for no further interr
        LCE     #15
        BSKZ    #A
        JMPC    ERROR
        LDX     RXVEX           ;restore volatile environment
        LDE     RXVEE
        LDA     RXVEA
        IONR    #F7
```

```
;
;
;
;  Receiver Interrupt Service Routine
;  ==================================
;
;  Amplitude decoding
;
RX30A:  LDA     BITS            ;update binary value
        LDE     RXWD2
        SLE
        STA     BITS
;
RX31A:  LDA     MBA             ;test against maximum
        SUB     IBA
        LCE     #04
        SKNZ
        JMPC    ERROR
        LDA     IBA
        INCA
        STA     IBA
;
        LDA     ACODE
        ADDC    #1
        ADD     IBA
        RAX
        ADD     MBA
        RAE
;
RX32A:  LDA     BITS            ;test binary value
        SUBW
        SKN
        JMPC    RX14
        REX
        ADDW
        STA     IA1
        CLRA
        STA     BITS
        STA     IBA
;
RX33A:  LDA     W3              ;write index to rx buffer
        ADD     B3
        RAX
        LDA     IA1
        BSET    #E              ;set amplitude tag
        STAW
;
RX34A:  LDA     W3              ;increment pointer
        INCA
        SUB     L3
        SKZP
        ADD     L3
        STA     W3
;
RX35A:  LDA     W3              ;test for overflow
        SUB     R3
```

```
        LCE     #02
        SKNZ
        JMPC    ERROR
;
        LDA     IAW1            ;increment word count
        INCA
        SUB     NAW
        SKZP
        ADD     NAW
        STA     IAW1
        SKZ
        JMPC    RX14
;
RX36A:  LDA     FLAG
        BCLR    DICT
        STA     FLAG
        JMPC    RX14
```

```
;
;
;
;  Receiver Interrupt Service Routine
;  ==================================
;
;  Tes Decoding
;
RX30T:  LDA     BITS            ;update binary value
        LDE     RXWD2
        SLE
        STA     BITS
;
RX31T:  LDA     MBT             ;test against maximum
        SUB     IBT
        LCE     #05
        SKNZ
        JMPC    ERROR
        LDA     IBT
        INCA
        STA     IBT
;
        LDA     TCODE
        ADDC    #1
        ADD     IBT
        RAX
        ADD     MBT
        RAE
;
RX32T:  LDA     BITS            ;test binary value
        SUBW
        SKN
        JMPC    RX14
        REX
        ADDW
        STA     IT1
        CLRA
        STA     BITS
        STA     IBT
;
RX33T:  LDA     W3              ;store index in rx buffer
        ADD     B3
        RAX
        LDA     IT1
        STAW
;
RX34T:  LDA     W3              ;increment pointer
        INCA
        SUB     L3
        SKZP
        ADD     L3
        STA     W3
;
RX35T:  LDA     W3              ;test for full rx buffer
        SUB     R3
        LCE     #03
```

```
        SKNZ
        JMPC    ERROR
;
RX36T:  LDA     ITW1            ;increment word count
        INCA
        SUB     NTW
        SKZP
        ADD     NTW
        STA     ITW1
        SKZ
        JMPC    RX14
        LDA     FLAG
        BSET    DICT
        STA     FLAG
        JMPC    RX14
```

```
;
;
;
;  Main Synthesis Loop
;  ===================
;
LOOP:   LDA     W4              ;test for full sample buffer
        SUB     R4
        SKZF
        ADD     L4
        ADDC    #1
        SUB     L4
        SKNZ
        JMPC    LOOP
;
SN11:   LDA     IS              ;test for end of epoch
        SKNZ
        JMPC    SN20
;
SN12:   LDA     DM              ;fetch next synthetic sample
        ADD     IS
        RAX
        LDAW
        OTA     MX              ;scale sample
        LDA     A
        OTA     MY
        INA     MXYLS
        RAE
        INA     MXYMS
        SLE
        SLE
        SLE
        SLE
        RAE
;
SN13:   LDA     W4              ;store sample
        ADD     B4
        RAX
        STEW
;
        LDA     W4
        INCA
        SUB     L4
        SKZF
        ADD     L4
        STA     W4
;
SN14:   LDA     IS              ;increment sample count
        INCA
        SUB     NS
        SKZF
        ADD     NS
        STA     IS
;
        JMPC    LOOP
```

```
;
;
;
;  End of Epoch Routine
;  ====================
;
SN20:   LDA     ITW             ;test for end of frame
        SKNZ
        JMPC    SN30
;
        LDA     R3              ;read next word
        ADD     B3
        RAX
        LDAW
        LCE     #12
        BSKZ    #E              ;test tag
        JMPC    ERROR
        RAE
;
        LDA     R3              ;increment pointer
        INCA
        SUB     L3
        SKZP
        ADD     L3
        STA     R3
;
        LDA     TCODE           ;fetch next index
        ADDC    #2
        ADD     MBT
        ADD     MBT
        ADDE
        RAX
        LDEW
        STE     NS
        ADD     NT
        RAX
        LDAW
        ADD     TCODE
        STA     DM
;
SN24:   CLRA                    ;reverse polarity
        STA     IS
        SUB     A
        STA     A
;
        LDA     ITW             ;increment Tes word count
        INCA
        SUB     NTW
        SKZP
        ADD     NTW
        STA     ITW
;
        JMPC    SN12
```

```
;
;
;
;  Standard End of Frame Routine
;  =============================
;
SN30:   LDA     PR              ;Load E reg. with delay pointer
        ADD     BR
        RAX
        LDEW
;
        LDA     W3              ;test for empty rx buffer
        SUB     R3
        SKZF
        ADD     L3
        SUB     NAW
        SUB     NTW
        SKZF
        STE     R3
;
        LDA     PR
        ADD     BR
        RAX
        LDA     R3
        STAW
;
        LDA     PR              ; Increment delay pointer
        INCA
        SUB     LR
        SKZF
        ADD     LR
        STA     PR
;
SN31:   LDA     R3              ;fetch amplitude index
        ADD     B3
        RAX
        LDAW
        LCE     #12
        BSKO    #E              ;check for tag present
        JMPC    ERROR
        BCLR    #E
        STA     IA
;
SN32:   LDA     R3              ;increment pointer
        INCA
        SUB     L3
        SKZF
        ADD     L3
        STA     R3
;
SN34:   LDA     ACODE           ;look up amplitude
        ADDC    #2
        ADD     MBA
        ADD     MBA
        ADD     IA
        RAX
```

```
        LDAW
        STA     A
;
        JMPC    SN21
        .END
```

Appendix 2

Variable length Codes

Variable length codes have been suggested as a means of reducing the delay associated with matching the TES source with a constant rate channel [37]. However, the codes suggested did not satisfy the prefix condition [98]. In response to the need, within the digital voice channel, for a method of constructing and implementing a more general class of binary variable length codes which satisfy the prefix condition and are easily decodable, the following method was developed by P.S. Cooper [52].

Let n_k denote the number of codewords of a given finite dictionary which are of length k (k is less than or equal to the maximum word length, K). If two codes with identical sets of n_k's are considered considered equivalent then a representative from each equivalent class can be selected to enable the same efficient decoding algorithm to be used for all codes. Note that if a code $(n_1, n_2, \ldots, n_K)$ is uniquely decodable then there is an equivalent prefix condition code. The representative codes are constructed as follows:

(1) Draw a binary tree of depth K.

(2) Label nodes such that left branches correspond to 0 and right branches to 1.

(3) set k = 1.

(4) At level k, starting from the left allocate the first n_k nodes to codewords.

(5) Delete those branches emanating from the codewords

of step 4

(6) Increase k by 1 and repeat steps 4 and 5 until all the codewords have been allocated.

Figure A2.1 shows the construction of the code (0,1,3,4,4). There will always be enough nodes for this process to continue as long as the n_k's satisfy the Kraft inequality.

$$\sum_{1}^{K} n_k . 2^{-k} \quad .LE. \quad 1$$

Constant length codes are included in this representation by sets of the form $(0,0, \ldots ,n_k)$ where there are n_k $(.LE. 2^K)$ codes of length K.

In terms of the binary values of the codewords it will be seen that they are arranged in numerical order and for codewords of the same length the binary values are consecutive. It is these two properties that make the codes easily decodable. Since all the decoder need do, having received k bits, is to compare the binary value of the received bits, R(k), with the maximum binary value, $\sum_2(k)$, for words of length k. If R(k) .GE. $\sum_2(k)$ then more bits are needed to decode the word. If R(k) .LT. $\sum_2(k)$ then $R(k) - \sum_2(k) + \sum_1(k)$ gives the position of the decoded word in the numerically ordered alphabet, where $\sum_1(k)$ is the position of the highest codeword of length k. Reflection on the way in which the codes were constructed yields the formula:

$$\sum_1(k) = n_1 + n_2 + \ldots + n_k$$

$$\sum_2(k) = 2^{k-1} n_1 + 2^{k-2} n_2 + \ldots + n_k$$

or recursively

$$\sum_1(k) = \sum_1(k-1) + n_k \quad , \quad \sum_1(0) = 0$$

$$\sum_2(k) = 2.\sum_2(k-1) + n_k \quad , \quad \sum_2(0) = 0$$

Since these two sums are functions of the n_k's, which are fixed, they need to be calculated once only and stored in suitable arrays. Table A2.1 shows these sums for the code of figure A2.1.

k	n_k	$\sum_1(k)$	$\sum_2(k)$	Codes
1	0	0	0	
2	1	1	1	00(0)
3	3	4	5	010(2),011(3),100(4)
4	4	8	14	1010(10),1011(11), 1100(12),1101(13)
5	4	12	32	11100(28),11101(29), 11110(30),11111(31)

Table A2.1 : Summations for decoding the numerically ordered prefix code (0,1,3,4,4).

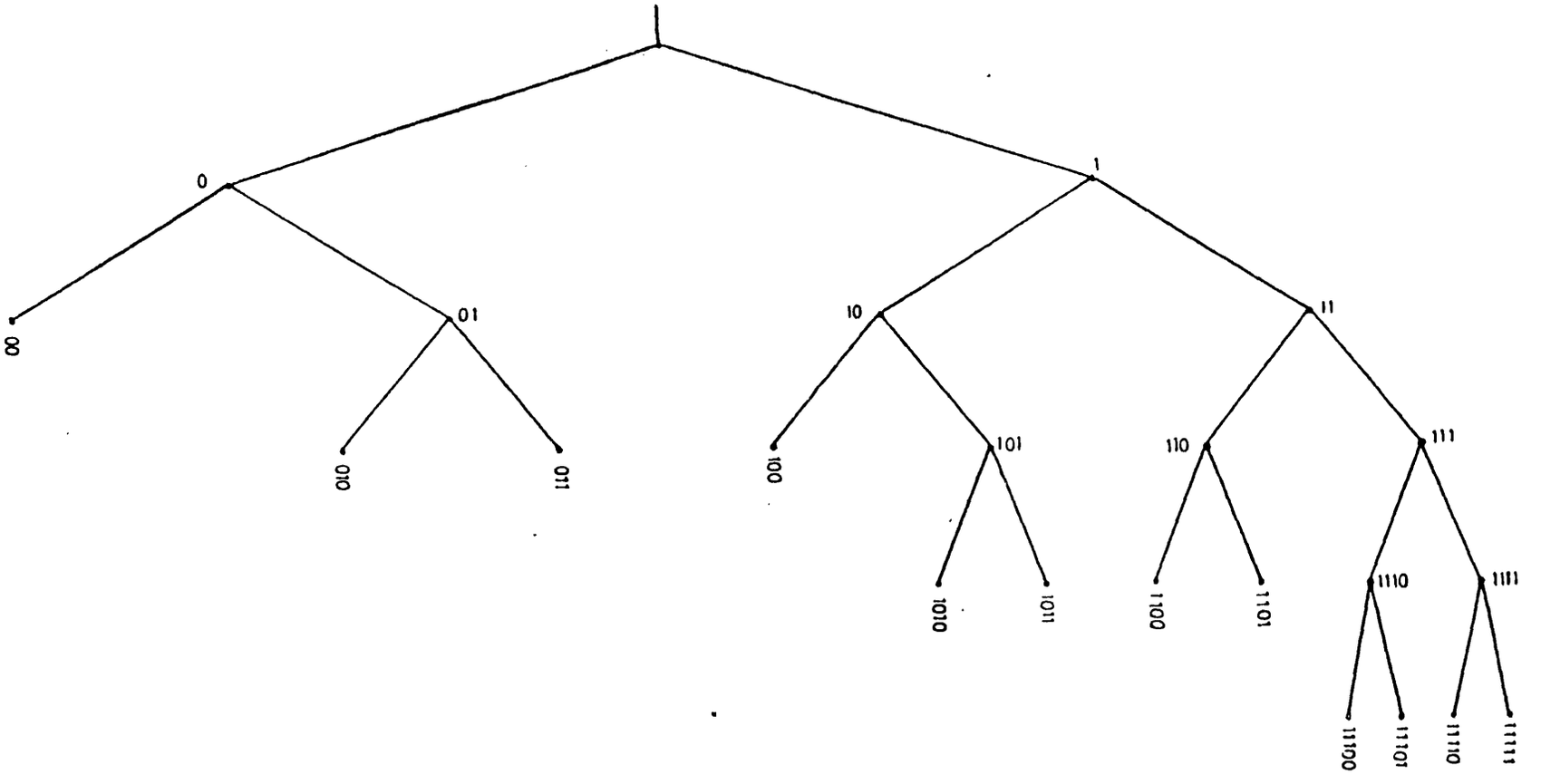


Figure A2.1 : Tree construction for numerically ordered prefix code (0,1,3,4,4)

Appendix 3

System Diagrams

To solve any problem computer programmers must perform a number of tasks in order to develop the required software for computation of that problem. These tasks are generally performed in a prescribed sequence and are listed below.

a) Problem Definition

b) Analysis and Flowcharting

c) Translation

d) Computation

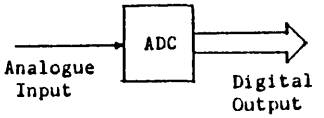
In high level language programming flow charts are employed for documentation of a program and, in general, are very satisfactory. However, microprocessor programming is different from high level programming in that relatively small programmes are written for particular sets of hardware, and commonly in assembler language.

Conventional flow charts which utilise boxes and lines have a number of weaknesses as a working document for assembly language programming. In order to overcome these weaknesses, alternative forms of flow charting have been proposed [99]. These techniques, however, still do not always produce a clear and compact problem representation which allows one to readily assess the overall function of a program. This is especially so for real-time processing where a microprocessor has been programmed to simulate a custom designed digital system.

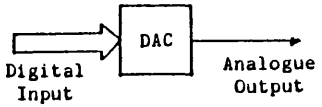
In an attempt to overcome these problems and to produce a compact representation of the programme, an alternative method of flow charting was developed by the author for the representation of programmes for real-time processing. This representation has been termed a SYSTEM DIAGRAM because it has the appearance of a circuit diagram instead of a conventional flow chart.

A set of standard symbols have been developed for the representation of data conversion, storage, arithmetic and logical functions which are software controlled or executed. The symbols are connected by either data buses or control lines or both. All data buses and stores may be allocated variable names which aids translation into assembly language. The system diagram may also be segmented with each segment being labelled. This also aided translation into the assembler language and indicates the segments functional priority.

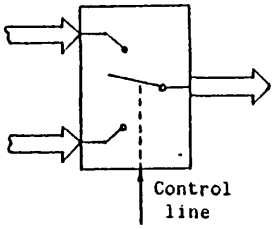
Unfortunately, the system diagrams do not convey information concerning the segments sequence timing and, in general, a short 'cover note' is required to compensate for this deficiency. Table A3.1 lists the symbols which have been developed and utilised within this thesis.



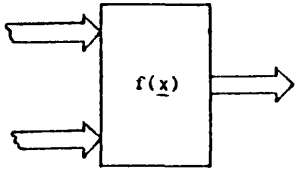
Analogue-to-Digital Converter



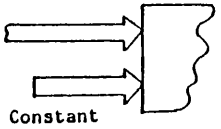
Digital-to-analogue Converter



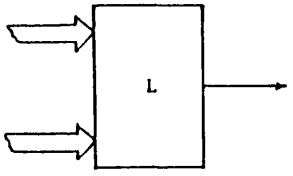
Multiplexer



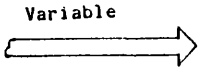
Arithmetic Function of Arithmetic Inputs



"Floating Input" - Input of set amplitude



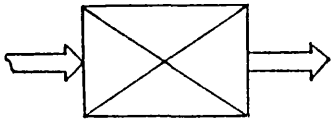
Logical Function of Arithmetic Inputs



Digital Variable

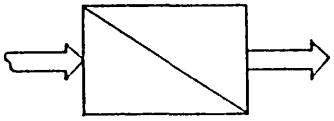
Control - Logical

Control - Logical; but not executed every clock pulse

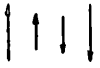


Data Buffer
First in, First out (FIFO)

One Step



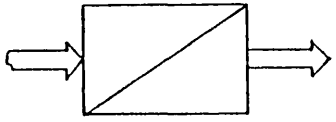
Address structured
Write only



Two Step

Z.C

Zero Crossing Detector



Address structured
Read only

C

Comparator(s)

Figure A3.1 : System diagram symbols and meanings.

Appendix 4

Supplementary Notes for System Diagrams of Section 3.3

The Miproc 16F processor and peripherals were structured such that the analogue to digital converter was triggered from an external clock. The 'end-of-conversion' signal being used to generate an interrupt to the processor.

The software was designed such that the processor "idled" until an interrupt occurred. The processor then serviced the INTERRUPT SERVICE ROUTINE (ISR).

The assembly code of the initial section of the Interrupt Service Routine (ISR) was identical for all the programmes. Once an interrupt had been generated, data was input from the Analogue to Digital Converter (ADC) and OFFSET subtracted, the result being stored in NEW. The value of OFFSET was set to eliminate the d.c. offset introduced by the systems hardware. The value of RESULT was then output to the Digital-to-Analogue Converter (DAC).

The following descriptions continue on from where the previous paragraph ended.

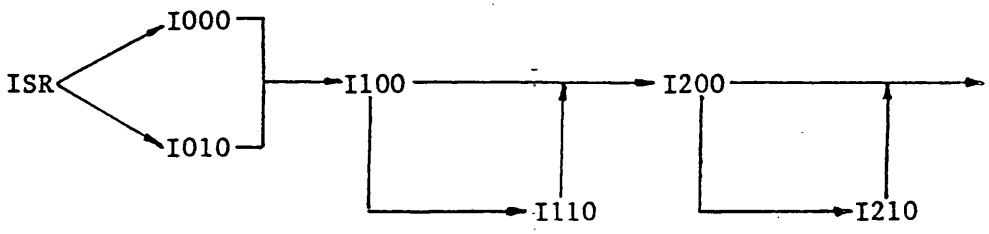
EAMP00 (Figure 3.2)

A value for OLD was output from BUFF and compared with its previous value for zero-crossing detection. If this had occurred a

a new value for PEAKO was output from XBUFF. OLD was then divided by PEAKO, the result of which was multiplied by LEVEL and the resultant stored in RESULT.

The value of NEW was input into BUFF and compared with its previous value for zero-crossing detection. If a zero-crossing had occurred, the amplitude parameter being detected, PEAKN, was input to XBUFF, and PEAKN was initialised.

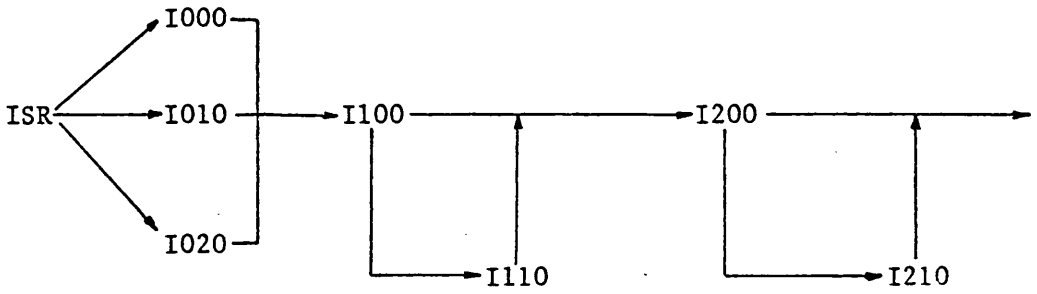
The inputs OFFSET and LEVEL were multiplexed, the control signal being the interrupt signal. The order of execution of the sub-sections within the ISR was:



EAMP10 (Figure 3.3)

This group of programmes were identical to the EAMP00 series except for the insertion of a threshold. When a value of PEAKO was output from XBUFF, it was compared with THRESH and if THRESH .GE. PEAKO, RESULT was set equal to zero.

The inputs OFFSET, LEVEL and THRESH were multiplexed, the control signal being the interrupt signal. The order of execution of the sub-sections within the ISR was:



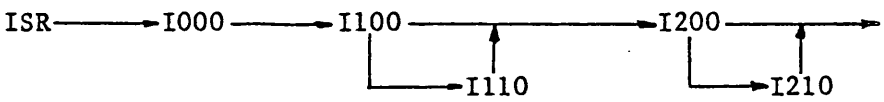
EAMP20 (Figure 3.4)

A value for OLD was output from BUFF and compared with its previous value for zero-crossing detection. If a zero-crossing had occurred a new value for PEAKI was output from IBUFF and a counter incremented. If this was the Nth zero-crossing a value for PEAKN was output from NBUFF and the counter initialized.

OLD was divided by PEAKI, multiplied by PEAKN and the resultant stored in RESULT.

The value of NEW was input to BUFF, and compared with its previous value for zero-crossing detection. If a zero-crossing had occurred, the amplitude parameter of the epoch, MAXI, was input to IBUFF and a counter incremented. If this was the Nth zero-crossing, the amplitude parameter for the group of epochs, MAXN, was input to NBUFF and the counter initialized.

The order of execution of the sub-sections within ISR was:



EAMP30 (Figure 3.5)

A value for OLD was outut from BUFF and compared with its previous value for zero-crossing detection. If a zero-crossing had occurred a new value for AMP was output from XBUFF, and the following conditions tested for and action taken.

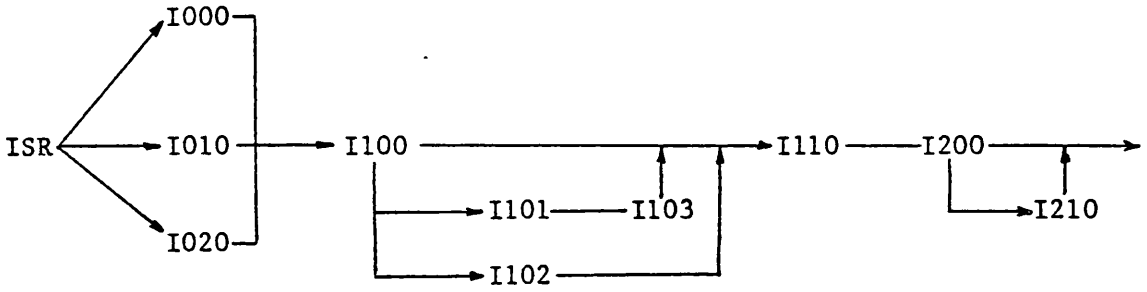
```
if      DAMP(n-1) .NE. 0  and  AMP .GE. DAMP(n-1)
then            DAMP(n) = DAMP(n-1) + STEP
otherwise               = DAMP(n-1) - STEP

if      DAMP(n-1) .EQ. 0  and  AMP .GE. THRESH
    then            DAMP(n) = STEP
    otherwise               = 0
```

where DAMP(n-1) and DAMP(n) were the previous and current processed amplitude parameters, respectively.

OLD was divided by AMP, multiplied by DAMP and the resultant stored in RESULT.

The value of NEW was input to BUFF and compared with its previous value for zero-crossing detection. If a zero-crossing had been detected, the amplitude parameter for the epoch, MAX, was input to XBUFF.

The inputs OFFSET, THRESH and STEP were multiplexed, the control signal being the interrupt signal. The order of execution of the sub-sections within the ISR was:



EAMP50 (Figure 3.6)

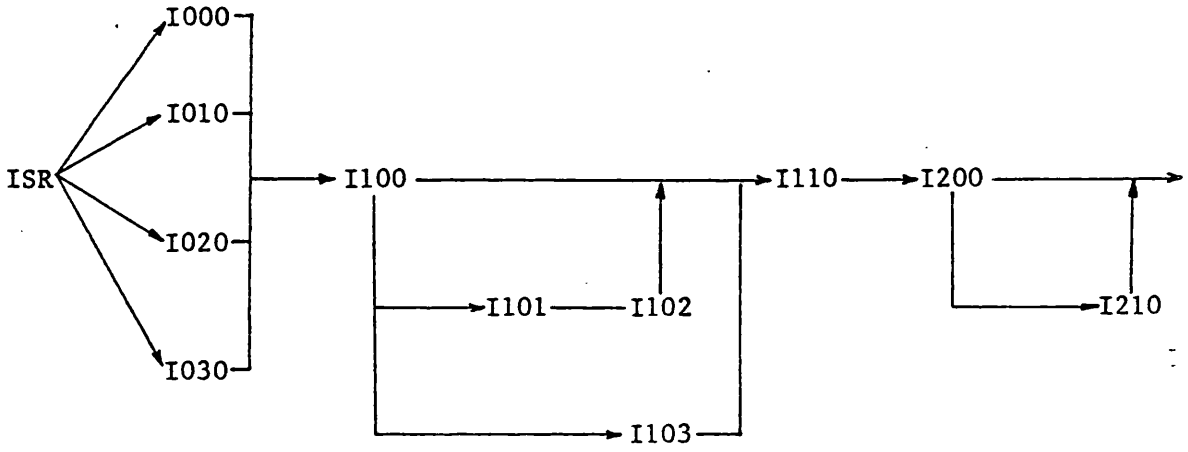
A value for OLD was output from BUFF and compared with its previous value for zero-crossing detection. If a zero-crossing had occurred, a new value for AMP was output from XBUFF, and the following conditions tested for and action taken.

```
If       DAMP(n-1) .NE. 0, and AMP .LT. DAMP(n-1)
and     |DAMP(n-1) + STEPL - AMP| .GE. |DAMP(n-1) + STEPH - AMP|
then       DAMP(n) = DAMP(n-1) + STEPL
otherwise          = DAMP(n-1) + STEPH
if       DAMP(n-1) .NE. 0, and AMP .LE. DAMP(n-1)
and     |DAMP(n-1) - STEPL - AMP| .LE. |DAMP(n-1) - STEPH - AMP|

then       DAMP(n) = DAMP(n-1) - STEPL
otherwise          = DAMP(n-1) - STEPH
if  DAMP(n-1) = 0, and AMP .LT. THRESH
then       DAMP(n) = 0
otherwise          = STEPL
```

where DAMP(n-1) and DAMP(n) were the previous and current processed amplitude parameters, respectively. OLD was divided by AMP, multiplied by DAMP and the resultant stored in RESULT. The value of

NEW was input to BUFF and compared with its previous value for zero-crossing detection. If a zero-crossing had been detected, the amplitude parameter for the epoch, MAX, was input to XBUFF.

The inputs OFFSET, THRESH, STEPL and STEPH were multiplexed, the control signal being the interrupt signal. The order of execution of the sub-sections within the ISR was:



Appendix 5

Walsh Functions

A5.1 Introduction

It is the purpose of this appendix to introduce Walsh functions [100] which form a complete orthogonal set of rectangular waves. Orthogonal means that if any two functions were multiplied together and integrated over the interval (sum the values) the resultant is zero unless the two functions were the same. Walsh functions are generally classified into three groups. These groups differ from one another in that the 'order' in which individual functions appear are different. The three types of order are :

(1) Sequency or Walsh ordering

(2) Paley or Dyadic ordering

(3) Normal or Hadamard ordering

In what follows, we define sequency and Rademacher functions and develop the Walsh functions based on Rademacher functions, Gray code and Hadamard matrices. Section A5.5 presents the Fortran sub-routine implemented in the investigations reported in chapter 6.

A5.2 Sequency

The term frequency is applied to a set of sinusoidal (periodic) functions whose zero-crossings are uniformly spaced over an interval. The generalisation of frequency is achieved by defining 'generalised

frequency' as one half the average number of zero-crossings per unit time [89]. Harmuth [101] introduced the term sequency to describe generalised frequency and applied it to distinguish functions whose zero-crossings are not uniformly spaced over an interval and which are not necessarily periodic. Analogous to frequency, which is expressed in Hertz or cycles per second, sequency is expressed in terms of Zero-Crossings per Second, generally abbreviated to "Zps".

A5.3 Rademacher Functions

Rademacher functions are an incomplete set of orthogonal functions which were developed in 1922 [102]. The Rademacher function of index n, denoted by R(n,t), is a train of rectangular pulses with 2^{n-1} zero-crossings over a normalised time base, that is 0 .LE. t .LE. 1, taking the values +1 and -1. An exception is R(0,t) which is a step function. The Rademacher functions may be generated using the following recursive relation:

$$R(n,t) = R(1,2^{n-1}t) \qquad \text{(A5.1)}$$

with $R(0,t) = 1$

and $$R(1,t) = \begin{cases} 1, & 0 \text{ .LT. } t \text{ .LE. } 0.5 \\ -1, & 0.5 \text{ .LT. } t \text{ .LE. } 1 \end{cases}$$

The first six Rademacher functions are given in figure A5.1

A5.4 Walsh Functions

The incomplete set of Rademacher functions were completed by Walsh [100] in 1923 to form the complete orthonormal set of rectangular functions, taking only two amplitude values +1 and -1, now known as Walsh functions.

(i) Sequency or Walsh Ordering

This is the ordering which was originally employed by Walsh [100]. We denote the set of walsh functions belonging to this set by:

$$S_w = Wal_w(n,t) , \quad n = 0,1,2, \ldots ,N-1 \qquad (A5.2)$$

where $N = 2^m$, $m = 1,2,3, \ldots$

The subscript "w" denotes Walsh ordering, n denotes the n-th member of S_w and t is the independant variable.

To calculate $Wal_w(n,t)$

(a) Write n in binary

(b) Convert n to Gray code

(c) Multiply together all Rademacher functions whose subscripts correspond to the position of the 1-bits in the Gray code number. They align as shown below:

Codeword	$g_n, \ldots ,g_2,g_1$
Rademacher	$R_n, \ldots ,R_2,R_1$

Using the above, the first eight Walsh functions have been

calculated and are shown in figure A5.2(a). Inspection of this diagram reveal that the sequency of a Walsh function is greater than or equal to that of the preceding Walsh function and has exactly one more zero-crossing. Hence the alternate name "Sequency ordering".

Sampling of the Walsh functions of Figure A5.2(a) at eight equi-distant points results in an (8×8) matrix, figure A5.2(b). Such matrices are denoted by $H_w(N)$, $N = 2^n$, since they may be achieved by re-ordering the rows of a class of matrices called Hadamard matrices.

(ii) Paley or Dyadic Ordering

This type of ordering was introduced by Paley [104]. In Paley's definition of Walsh functions, their sequencies are arranged in Gray code where the Gray code is the natural way of ordering binary vectors in dyadic space. Hence the alternate name "Dyadic Ordering". This set of Walsh functions are denoted by:

$$S_p = Wal_p(n,t) , \qquad n = 0,1,2, \ldots ,N-1$$

Where the subscript "p" denotes Paley ordering. The set S_p is related to the Walsh ordered set, S_w, by the relation:

$$Wal_p(n,t) = Wal_w(g[n],t) \qquad (A5.3)$$

Where g[n] represents the Gray-to-Binary conversion of n. The first eight Paley ordered functions are given in figure A5.3(a). Sampling the functions of figure A5.3(a), at eight equi-distant points, we achieve the (8×8) matrix of figure A5.3(b). Once more

this matrix may be derived by re-arranging the rows of the (8×8) Hadamard matrix. The matrices associated with the Paley ordered Walsh functions are denoted by $H_p(N)$, $N = 2^n$.

(iii) Natural or Hadamard Ordering

This set of Walsh functions are denoted by:

$$S_h = Wal_h(n,t) , \qquad n = 0,1,2, \ldots ,N-1$$

Where the subscript "h" denotes Hadamard ordering. The functions of S_h are related to the Walsh functions by the relation:

$$Wal_h(n,t) = Wal_w(g[|n|],t) \qquad (A5.4)$$

Where $|n|$ is achieved by the bit reversal of n

and $g[|n|]$ is the Gray-to-Binary conversion of $|n|$.

Figure A5.4(a) gives the first eight Hadamard ordered Walsh functions. Sampling these functions results in the (8×8) Hadamard matrix of figure A5.4(b). In general an (N×N) matrix $H_h(N)$, $N = 2^n$, would be obtained. Hadamard matrices may also be generated recursively from:

$$H_h(1) = 1$$

and

$$H_h(2m) = \begin{vmatrix} H_h(m) & H_h(m) \\ H_h(m) & -H_h(m) \end{vmatrix} \qquad m = 1,2, \ldots \qquad (A5.5)$$

$$= H_h(1) \bullet H_h(m) \qquad (A5.6)$$

Where the symbol $\bullet$ denotes the Kronecker product.

The Hadamard matrix is both symmetrical and orthogonal ie. $H_h(N).H_h(N) = N.I(N)$ where I(N) is an (N×N) identity matrix. Hadamard matrices in the form of equation A5.5 are considered to be in "natural" form [82], hence the name natural ordering.

A5.5 Transformation Subroutine

There are several algorithms of varying complexity and memory requirements for computing the discrete Walsh or Hadamard Transform [86,104-106]. The subroutine utilised for the computation of the Fast Walsh Transform is given in Listing A5.1 and that employed for the Fast Hadamard Transform is given in Listing A5.2.

```
      SUBROUTINE FWT(N,M)
C
      COMMON    IX
      DIMENSION IX(1024)
C
C     Subroutine for calculation of a
C     Fast 512 point Sequency ordered
C     Walsh Transform.
C
      NH = N/2
      L1 = 0
      DO 300 L = 1,M
      L1 = N - L1
      L2 = N - L1
      N1 = 0
      N3 = 2**LM
      N3I = 2*N3
      N3N = N/N3I
      DO 200 I = 1,N3N
      N2 = N1 + 1
      N1 = N1 + N3
      JS = (I - 1)*N3I
      JD = JS + N3I + 1
      DO 100 J = N2,N1
      JS = JS + 1
      J2 = J + NH
      III = L2 + J
      JJJ = L2 + J2
      IX(L1 + JS) = IX(III) + IX(JJJ)
      JD = JD + 1
      IX(L1 + JD) = IX(III) - IX(JJJ)
100   CONTINUE
200   CONTINUE
300   CONTINUE
      IF (L1) 400,600,400
400   DO 500 I = 1,N
      II = I + N
      IX(I) = IX(II)
500   CONTINUE
600   RETURN
      END
```

Listing A5.1 : Fast Walsh Transform Subroutine.

```
      SUBROUTINE FHT(NN,M)
C
      COMMON
      DIMENSION IX(512)
C
C     Subroutine for the calculation of a
C     Fast 512 point Hadamard ordered
C     Walsh Transform.
C
      NO = NN/2
10    J = 0
20    K = J + NO
30    J = J + 1
      I = J + NO
      IT = MR(J) - MR(I)
      MR(J) = IT
      IF (J - K) 30,40,40,
40    J = J + NO
      IF (J - NN) 20,50,50
50    NO = NO/2
      IF (NO) 60,60,10
60    RETURN
      END
```

Listing A5.2 : Fast Hadamard Transform Subroutine.

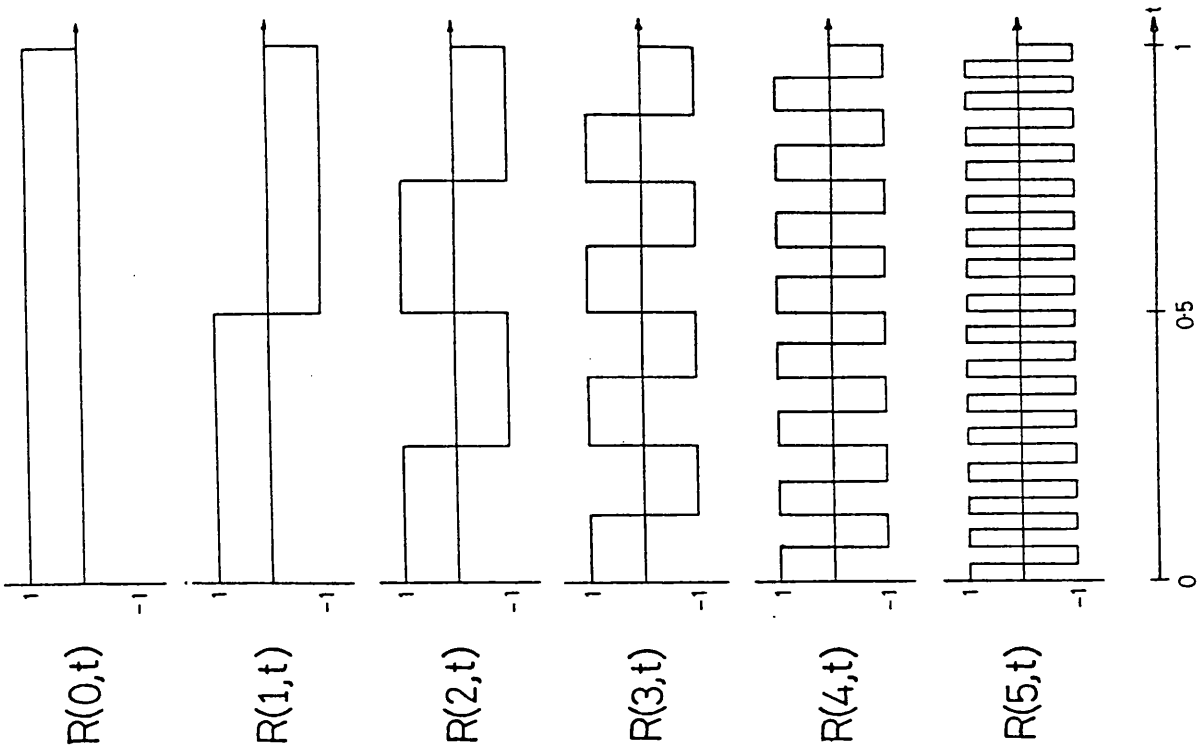
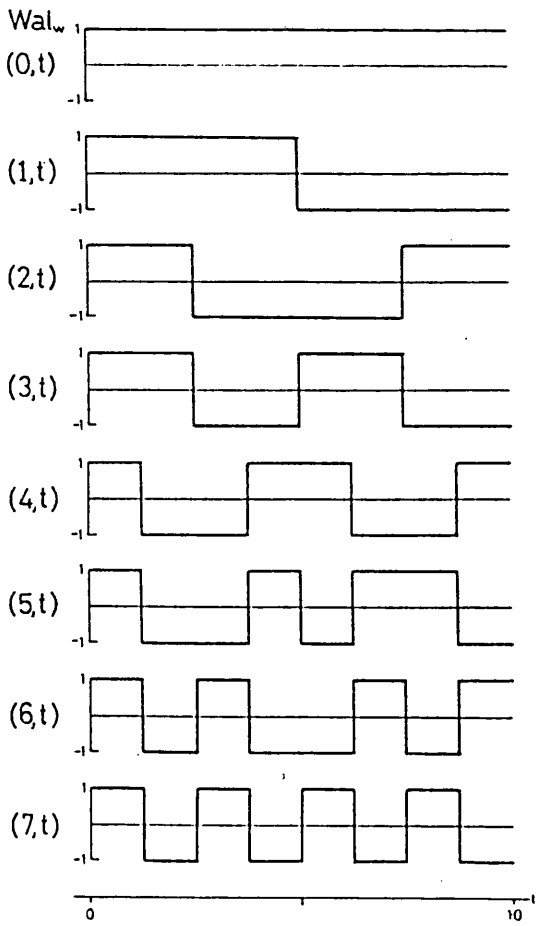


Figure A5.1 : The first six Rademacher functions

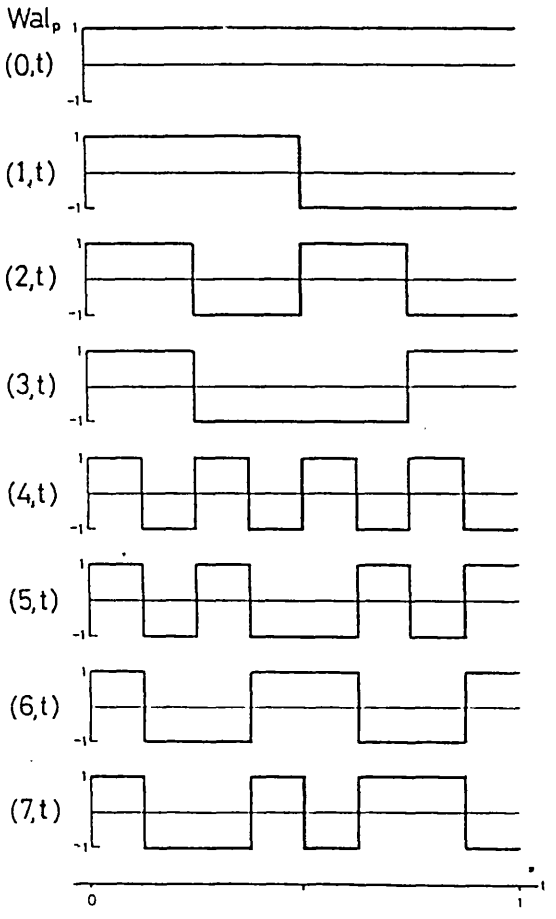


$$H_w(8) = \begin{bmatrix} + & + & + & + & + & + & + & + \\ + & + & + & + & - & - & - & - \\ + & + & - & - & - & - & + & + \\ + & + & - & - & + & + & - & - \\ + & - & - & + & + & - & - & + \\ + & - & - & + & - & + & + & - \\ + & - & + & - & - & + & - & + \\ + & - & + & - & + & - & + & - \end{bmatrix}$$

Figure A5.2 : Sequency Ordered Walsh Functions

(a) Continuous, N = 8.

(b) Discrete, N = 8.

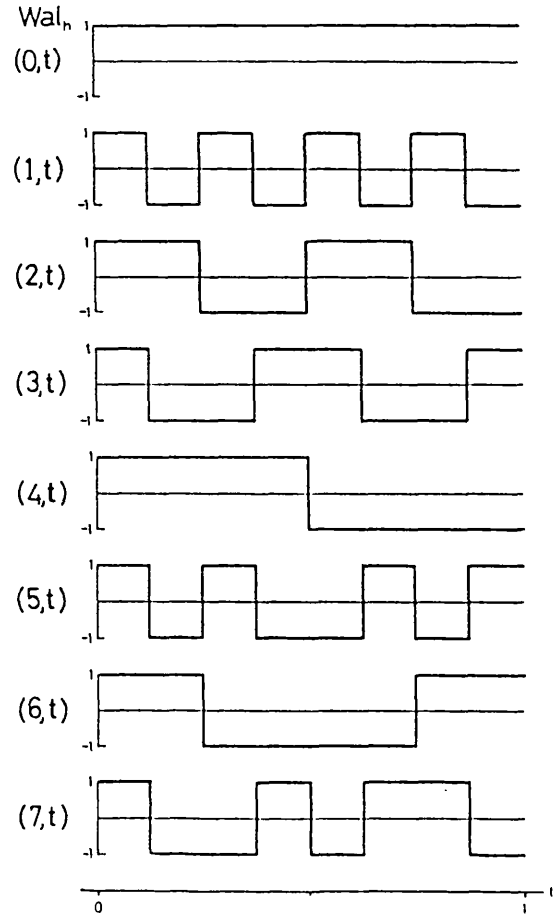


$$H_p(8) = \begin{bmatrix} + & + & + & + & + & + & + & + \\ + & + & + & + & - & - & - & - \\ + & + & - & - & + & + & - & - \\ + & + & - & - & - & - & + & + \\ + & - & + & - & + & - & + & - \\ + & - & + & - & - & + & - & + \\ + & - & - & + & + & - & - & + \\ + & - & - & + & - & + & + & - \end{bmatrix}$$

Figure A5.3 : Paley Ordered Walsh Functions

(a) Continuous, N = 8.

(b) Discrete, N = 8.



$$H_h(8) = \begin{bmatrix} + & + & + & + & + & + & + & + \\ + & - & + & - & + & - & + & - \\ + & + & - & - & + & + & - & - \\ + & - & - & + & + & - & - & + \\ + & + & + & + & - & - & - & - \\ + & - & + & - & - & + & - & + \\ + & + & - & - & - & - & + & + \\ + & - & - & + & - & + & + & - \end{bmatrix}$$

Figure A5.4 : Hadamard Ordered Walsh Functions

(a) Continuous, N = 8.

(b) Discrete, N = 8.

Appendix 6

Physical Significance of Discarding Walsh Transform Coefficients

A6.1 Hadamard Ordered Coefficients

Any process which involves transforming to the Walsh domain using a Hadamard Ordered Walsh Transformation may be represented as follows:

$$\underline{x}^{*} = H_h^{-1}(N).P(N).H_h(N).\underline{x} \qquad (A6.1)$$

where $\underline{x}$: is a column vector containing the original data.

$H_h(N)$: is the forward Hadamard Transform matrix.

$H_h^{-1}(N)$: is the inverse Hadamard Transform matrix.

$P(N)$: is the process operator, which for undistorted transformation is an (N×N) unity matrix, and therefore usually omitted.

$\underline{x}^{*}$: is a column vector containing the new data.

The elements of P(N) can thus be dictated to define operations within the transform domain. In the following sections we shall examine two structures of P(N) and their physical significance on the resulting data.

(i) Discarding the last N/2 coefficients of an N point transform, where $N = 2^n$, $n = 1,2,3, \ldots$

Let $m = N/2$

then
$$P(N) = \begin{vmatrix} I(m) & \underline{0} \\ \underline{0} & \underline{0} \end{vmatrix}$$

Where P(N) has been partitioned into (m×m) matrices and I(m) is an identity matrix. Since $H_h^{-1}(N)$, P(N) and $H_h(N)$ are of the same order they may be combined into a single matrix. The inverse Hadamard Ordered Walsh Transform is defined as [83]:

$$H_h^{-1}(N) = \frac{1}{N} H_h(n)$$

and from the recursive relationship (appendix 5, equation A5.5) it is known that:

$$H_h(2N) = \begin{vmatrix} H_h(N) & H_h(N) \\ H_h(N) & -H_h(N) \end{vmatrix}$$

therefore

$$H_h^{-1}(N).P(N).H_h(N) = \begin{vmatrix} H_h(m) & H_h(m) \\ H_h(m) & -H_h(m) \end{vmatrix} \begin{vmatrix} I(m) & \underline{0} \\ \underline{0} & \underline{0} \end{vmatrix} \begin{vmatrix} H_h(m) & H_h(m) \\ H_h(m) & -H_h(m) \end{vmatrix}$$

$$= \begin{vmatrix} H_h(m).H_h(m) & H_h(m).H_h(m) \\ H_h(m).H_h(m) & H_h(m).H_h(m) \end{vmatrix}$$

However $$H_h(m).H_h(m) = m.I(m)$$

thus $$H_h^{-1}(N).P(N).H_h(N) = \frac{1}{2}\begin{vmatrix} I(m) & I(m) \\ I(m) & I(m) \end{vmatrix}$$

eg. for N = 4

$$H_h^{-1}(N).P(N).H_h(N) = \frac{1}{2}\begin{vmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{vmatrix}$$

The physical significances of this matrix are:

(a) The column vector of N new data samples is comprised of the the first N/2 new samples and their repeat.

(b) The N/2 new data samples may be calculated thus

$$x_n^* = x_{(n+N/2)}^* = \frac{x_n + x_{(n+N/2)}}{2} \quad \text{for } n = 1,2, \ldots ,N/2$$

and (c) From (b) above, it is obvious that N may be any even value greater than 2 and not necessarily a power of 2.

(ii) Discarding alternate coefficients of an N point transform.

In this case P(N) will have alternate 1's and 0's along the leading diagonal and all other elements will be zero.

eg. for N = 4

$$P(4) = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

Due to the structure of P(N), the matrices $H_h^{-1}(N)$, P(N) and $H_h(N)$ cannot be combined, and simply represented, using identity matrices. Therefore $H_h^{-1}(N)$ and $H_h(N)$ are represented below by their elements, H_{ij}, for the case of N = 4.

$$H_h^{-1}(4).P(4).H_h(4) = \frac{1}{4} \begin{vmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{vmatrix} \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{vmatrix} \begin{vmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{vmatrix}$$

$$= \frac{1}{2} \begin{vmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{vmatrix}$$

The physical significances of the resulting matrix are:

(a) The column vector, of N new data samples, is comprised of N/2 new samples, each of which is immediately followed by itself repeated.

(b) The N/2 new data samples may be calculated thus

$$x_n^* = x_{n+1}^* = \frac{x_n + x_{n+1}}{2} \quad \text{for } n = 1,3, \ldots ,N-1$$

(c) From (b) above, it is obvious that N can be any even value greater than 2 and not necessarily a power of 2.

A6.2 Walsh Ordered Coefficients

Any process involving transformations into the Walsh domain and employing a Walsh Ordered Walsh Transformation, may be represented as follows:

$$\underline{x}^{*} = H_{w}^{-1}(N).P(N).H_{w}(N) \qquad (A6.2)$$

where $H_{w}(N)$: is the forward Walsh Transform

$H_{w}^{-1}(N)$: is the inverse Walsh Transform

and the remainder are as stated in the previous section (A6.1)

(i) Discarding the last N/2 coefficients of an N point transform, where $N = 2^{n}$, $n = 1,2,3, \ldots$

By inserting the appropriate elements into the matrices of equation A6.2 it can be shown that this process is equivalent to the discarding of alternate coefficients of a Hadamard Ordered Walsh Transform (section A6.1(ii)).

(ii) Discarding alternate coefficients of an N point transform

It can be shown that $H_{w}^{-1}(N).P(N).H_{w}(N)$, for N = 4, reduces to

$$= \frac{1}{2}\begin{vmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{vmatrix}$$

The physical significances of this matrix are:

(a) The column vector of N new data samples is comprised of N/2 new data samples in the first N/2 rows, the remaining N/2 rows are a mirror image of the first N/2

(b) The N/2 new data samples may be calculated thus

$$x^*_n = x^*_{N+1-n} = \frac{x_n + x_{N+1-1}}{2} \qquad \text{for } n = 1,2, \ldots ,N/2$$

(c) From (b) it is noted that N may be any even value .GE. 2 and not necessarily a power of 2.

(d) If N = 2, we achieve the same result as for discarding alternate Hadamard ordered coefficients (section A6.1(ii)).

A6.3 Paley Ordered Coefficients

Any process, which involves transformation into the Walsh domain employing a Paley Ordered Walsh Transformation may be represented as follows:

$$\underline{x}^* = H_p^{-1}(N).P(N).H_p(N) \qquad \text{(A6.3)}$$

where $H_p(N)$: is the forward Paley Transform

$H_p^{-1}(N)$: is the inverse Paley Transform

and the remainder are as stated in the previous section (A6.1)

(i) Discarding the last N/2 coefficients of an N point transform, where $N = 2^n$, n = 1,2,3, ...

It can be proven that this process is equivalent to the discarding of alternate Hadamard ordered coefficients.

(ii) Discarding alternate coefficients of an N point transform

By inserting the (N×N) elements into the matrices of equation A6.3 it can be demonstrated that this process is equivalent to the discarding of the last N/2 coefficients of an N point Hadamard Ordered Walsh Transform.

A6.4 Dominant Coefficient Technique

In sections A6.1, A6.2 and A6.3 the resultant column vector of a Walsh Transformation has been predicted in terms of the input vector, after elimination of either alternate coefficients or the last N/2 coefficients before the inverse transform was performed. A technique employed by a number of researchers for achieving data reductions is that of Dominant Coefficient Retention [85,86,88]. An attempt to predict the resultant column vector for this technique has not been conducted since it is dependent on the sequency components embedded in the input data. Although the components affected cannot be predicted, it can be stated that, the output column vector would be the same irrespective of which Walsh transform was employed ie. Walsh, Paley or Hadamard. This is because the three transforms only differ in their sequency ordering and this would not have any effect on the resulting column vector.

Appendix 7

Speech Intelligiblity and Preference Assesment

A7.1 Introduction

Speech quality may be viewed as the totality of a communcication channels characteristics or the combination of the different attributes of speech which must be preserved to give the average listener a high quality voice signal. Speech quality can be said to be describeable in terms of four psychoacoustic attributes : intelligibility, preference, loudness and speaker recognition.

In the investigations reported in this thesis intelligibility and preference were considered the main attributes of speech to be considered for the evaluation of speech systems and loudness was considered a physical factor and speaker recognition was considered a psychological factor.

Except for intelligibility and preference, quantitative definitions of the physical and psychological factors have been impossible to obtain, with the result that an accurate definition of speech quality depends upon interpretation. It was for this reason that the IEEE subcommittee on subjective speech quality measurement [107] concluded "that a single method of subjective measurement of speech quality could not be recommended".

(i) Intelligibility

Intelligibility is a measure of the ability of a communication system to convey spoken information to an average listener. Although intelligibility measurements are well established the intelligibility scores achieved are not absolute quantities but are functions of:

- the listener
- speakers
- procedures
- the test material employed

Generally, intelligibility measurements evaluate systems in the gross sense, the number of words correct for a given test [108]. However, there have been measures developed directed towards the analysis of the system in terms of the factors contributing to intelligibility. The Diagnostic Rhyme Test (DRT) [109] is directed towards diagnostic testing of phonemic errors in the listeners response.

(ii) Preference

Preference can be thought of as an expression of the degree that one speech signal is preferred to another one, irrespective of the particular reasons of the listeners for their decisions. Preference answers the question : "how well does an average listener like a particular speech test signal as a source of information ?".

This question may be answered in two ways. Firstly by comparing the speech test signal consecutively with a variable speech reference signal (Isopreference test [107]) or secondly by rating his average attitude towards the signal alone (Direct Comparison Test, DCT).

The direct comparison of two speech signals is the most basic method for making preference decisions. However, it is the most direct approach to answering the question "which one of the two signals A and B is quantitatively better ?". This method does not yield a single absolute value.

In the following sections we shall briefly review the performance evaluating techniques employed during these investigations.

A7.2 Diagnostic Rhyme Test (DRT)

The Diagnostic Rhyme Test (DRT), developed by Voier [109] and simplified by Wong and Markel [110], is a special purpose intelligibility test that determines intelligibility and diagnoses what sounds the voice system does not transmit properly.

The DRT is a two choice test of consonant discriminability, which yields a gross measure of speech intelligibility and additional scores relating to the performance of the speaker, listener or system under test.

The reduced version of the DRT [110] utilises a collection of 96 words (46 rhyming pairs selected such that the initial conson-

ants of each pair differ in terms of a single phonemic attribute). One member of the rhyming pair acts as a stimulus and the listener's task is to indicate which member of the word pair was spoken. A correct choice indicates that the listener has, in effect, discriminated the state of one of six perceptual attributes of English consonant phonemes. In the event of being unable to identify, the listeners are requested to make a guess.

During a test each word pair is presented twice at random. On the second presentation the order of the rhyming pair is reversed, so that both words of the pair occur at a particular position, ie. as the first or second word once during the test.

Depending on the word pair involved, each stimuli serves to test the discriminability of one of the following six perceptual phonemic attributes:

- Voicing
- Nasality
- Sustention
- Sibilation
- Graveness
- Compactness

Table A7.1(a) indicates the features of speech for which each attribute is tested for discriminability. The word pairs employed are given in Table A7.1(b). In the Table, the positive state (eg. voiced) of each feature is represented in the left member of each pair; the negative state (eg. unvoiced) is represented in the right

member of each pair.

The results of the listeners responses are then presented as a gross percentage score for the discriminability of each of these attributes. Separate scores for the discriminability of each state of each attrbute may be obtained by appropriate analysis of the listeners reponses.

All percentage scores are calculated as follows to take into account the effects of chance:

$$D = \frac{R - W}{T} \quad 100\%$$

where D : is percent correct discriminations

R : is the number of correct responses

W : is the number of incorrect responses

and T : is the total number of responses.

A specimen of the scoring sheet employed in these tests is given in Table A7.2

The single word stimuli were recorded in an acoustically quiet room using a Sony microphone (ECM 170) and Sony cassette recorder (TC 158 cs).

The listening test sessions were conducted in a Tandberg language laboratory (156-B) which consisted of 20 double-walled listening booths. Listening was performed using Amplivox (Astro-lite) headphones.

A7.3 Direct Comparison Test (DCT)

The Direct Comparison Test (DCT) [57] of two signals is the most basic method of preference judgement. This method, however, does not yield a single absolute value of preference but merely indicates whether one of the two signals presented is preferable to the other.

In many cases this is the most convenient method of evaluating speech quality and provides a reasonable indication of the order of preference of a given set of speech signals.

The listeners are presented with two signals processed under different conditions and are required to indicate under which condition they would prefer to make a long conversation. In the event of the listeners not being able to make a choice, they are requested to indicate this as well so that none of the comparisons are left without a decision.

Each pair of signals are presented twice during a test and at the second presentation the recording order is reversed to eliminate any local effects. The two signals to be compared are presented consecutively and the pairs of signals are presented randomly, separated by three second intervals.

The tests were performed in an acoustically quiet room. The stimuli being persented to the listener via a JVC cassette recorder (CD1635 Mk II), Sony Amplifier (TA-70) and Sony Headphones (DR-9).

ATTRIBUTE	FEATURES TESTED
Voicing	For half of the words both critical phonemes involve friction. The remaining critical phonemes are stops.
Nasality	In half of the words, each vowel context involve a grave phoneme-pair (ie. /m - b/). The other half involve an acute pair (ie. /n - d/).
Sustention	Half of the words involver a voiced phoneme pair and half, an unvoiced phoneme pair.
Sibilation	Half of the words involve voiced phonemes and half involve unvoiced phonemes.
Graveness	The words were constructed such that for each vowel context, one word lies in the voiced "plane", one in the unvoiced; one in the sustained "plane", and one in thr interrupted.
Compactness	Words were constructed such that both states of voicing, sibilation and sustention were given equal representation, though not in all vowel contexts.

Table A7.1(a) : The features of speech for which each attribute of the DRT tests for discriminability [111].

VOICING: (Voiced - Voiceless)

1.	BEAN	PEEN
3.	GIN	CHIN
4.	ZED	SAID
5.	VAULT	FAULT
6.	GOAT	COAT

NASALITY: (Nasal - Oral)

7.	MOSS	BOSS
9.	KNOCK	DOCK
10.	MILT	BILT

SUSTENTION: (Continuant - Interrupted)

11.	SHEET	CHEAT
12.	FOOH	POOH
13.	SHOES	CHOOSE
14.	VILL	BILL
15.	THICK	TICK
16.	THOUGH	DOUGH
17.	THEN	DEN
18.	FENCE	PENCE
19.	THONG	TONG
20.	SHAW	CHAW
21.	SHAD	CHAD
22.	VON	BON

SIBILATION: (Strident - Mellow)

23.	ZEE	THEE
24.	JUICE	GOOSE
25.	CHAIR	CARE
26.	SANK	THANK
27.	SING	THING
28.	JOE	GO
29.	SAW	THAW
30.	ZEN	THEN

GRAVENESS: (Grave - Acute)

31.	FIN	THIN
32.	MOON	NOON
33.	BANK	DANK
34.	POT	TOT
35.	WEED	REED
36.	FOUGHT	THOUGHT
37.	POOL	TOOL
38.	MET	NET

COMPACTNESS: (Compact - Diffuse)

39.	GILL	DILL
40.	GHOUST	BOAST
41.	CAUGHT	TAUGHT
42.	SHAG	SAG
43.	KEG	PEG
44.	HIT	FIT
45.	YOU	RUE
46.	YIELD	WIELD

Table A7.1(b) : A reduced DRT word pair list

TEST NO 1

THEN	DEN	SHEET	CHEAT	WEED	REED	WEED	REED	THICK	TICK	VAULT	FAULT
THICK	TICK	CHAIR	CARE	JOE	GO	SHAW	CHAW	FOOH	POOH	NAB	DAB
SAW	THAW	CHAIR	CARE	FENCE	PENCE	SANK	THANK	CHAIR	CARE	ZED	SAID
WEED	REED	YIELD	WIELD	BANK	DANK	ZEE	THEE	MET	NET	POT	TOT
DUNE	TUNE	MILT	BILT	SANK	THANK	FOOL	TOOL	ZEE	THEE	FENCE	PENCE
GOAT	COAT	CAUGHT	TAUGHT	CAUGHT	TAUGHT	KNOCK	DOCK	FIN	THIN	MOSS	BOSS
GHOST	BOAST	HIT	FIT	SHAG	SAG	SHAW	CHAW	SANK	THANK	HIT	FIT
VILL	BILL	JUICE	GOOSE	GILL	DILL	KNOCK	DOCK	POOH	POOH	THEN	DEN
VILL	BILL	SAW	THAW	SHAG	SAG	DUNE	TUNE	FOOL	TOOL	MOON	NOON
CHAIR	CARE	FIN	THIN	MOSS	BOSS	CAUGHT	TAUGHT	YOU	RUE	FIN	THIN
ZEN	THEN	CAUGHT	TAUGHT	GILL	DILL	THOUGH	DOUGH	SHAD	CHAD	KEG	PEG
GHOST	BOAST	POOH	POOH	JOE	GO	JOE	GO	THONG	TONG	YOU	RUE
SHEET	CHEAT	SHOES	CHOOSE	MET	NET	SING	THING	YOU	RUE	BEAN	PEEN
THEN	DEN	MOON	NOON	KNOCK	DOCK	MOSS	BOSS	BANK	DANK	THOUGH	DOUGH
FOOL	TOOL	VILL	BILL	POT	TOT	YIELD	WIELD	KEG	PEG	SING	THING
YIELD	WIELD	MOON	NOON	THONG	TONG	THICK	TICK	MILT	BILT	SHEET	CHEAT
THONG	TONG	THICK	TICK	SING	THING	JUICE	GOOSE	HIT	FIT	SHEET	CHEAT
FIN	THIN	MILT	BILT	MET	NET	JOE	GO	BEAN	PEEN	VON	BON
ZEN	THEN	GIN	CHIN	MILT	BILT	SHAG	SAG	GHOST	BOAST	THEN	DEN
ZEN	THEN	THOUGH	DOUGH	SHAW	CHAW	SAW	THAW	SANK	THANK	FOUGHT	THOUGHT
HIT	FIT	SHAD	CHAD	ZED	SAID	SAW	THAW	BANK	DANK	SHAD	CHAD
ZEN	THEN	BANK	DANK	GIN	CHIN	VAULT	FAULT	FOUGHT	THOUGHT	YIELD	WIELD
JUICE	GOOSE	POT	TOT	WEED	REED	VON	BON	GOAT	COAT	SHAG	SAG
GHOST	BOAST	SHOES	CHOOSE	NAB	DAB	FOUGHT	THOUGHT	ZED	SAID	THONG	TONG
MET	NET	VON	BON	FENCE	PENCE	GILL	DILL	VILL	BILL	MOON	NOON
NAB	DAB	VAULT	FAULT	ZEE	THEE	KEG	PEG	SHOES	CHOOSE	VON	BON
SING	THING	ZEE	THEE	ZED	SAID	KEG	PEG	GIN	CHIN	POT	TOT
FOUGHT	THOUGHT	NAB	DAB	FOOL	TOOL	GILL	DILL	DUNE	TUNE	DUNE	TUNE
BEAN	PEEN	JUICE	GOOSE	SHAW	CHAW	MOSS	BOSS	GIN	CHIN	GOAT	COAT
VAULT	FAULT	FOOH	POOH	THOUGH	DOUGH	GOAT	COAT	KNOCK	DOCK		
SHOES	CHOOSE	FENCE	PENCE	SHAD	CHAD	YOU	RUE	BEAN	PEEN		

Table A7.2 : Reduced specimen of the DRT scoring sheet.

Subjective Listening Tests

You will hear a sequence of pairs of sentences and you should state which one (if any) of the pair you find more acceptable. So as to familiaries you with the procedure a practice pair is provided. Indicate your preference by placing a tick in the appropriate column adjacent to the pair you are listening to, e.g.

Pair No.	Codeword	Sentence 1	Sentence 2
5	Tango	✓	

Indicates preference of Sentence 1

If there is NO PREFERENCE please tick BOTH columns. The codeword gives an indication of the content of the sentence pair. This is to assist you in keeping track and marking the appropriate row.

Pair No.	Codeword	Sentence 1	Sentence 2
Practice Pair	Tango		
1	Tango		
2	Tango		
3	Oscar		
4	Tango		
5	Oscar		
6	Oscar		
7	Tango		
8	Tango		
9	Oscar		
10	Tango		
11	Oscar		
12	Oscar		
13	Tango		
14	Tango		

Table A7.3 : Specimen of the DCT scoring sheet.

Appendix 8

Speech Material

The speech material employed for the informal subjective appraisals of the speech synthesised by the various coders in this thesis are listed below. The utterances were supplied by Dr. J. N. Holmes formerly of the Joint Speech Research Unit, GCHQ, Cheltenham.

1: Male : A bird in the hand is worth two in the bush.

2: Male : An apple a day keeps the doctor away.

3: Male : Hello operator, operator.
Female : Yes, what can I do for you ?
Male : I'd like to make a telephone call to Ballem in England.
Female : Did you say Wallem in England ?
Male : No, I said Ballem in England.
Female : What part of England is that ?
Male : It's close to Newcastle.
Female : Have you got the area code ?
Male : No I'm affraid not. I want Ballem 64125.
Female : Is it a personnal call ?
Male : Yes, I'd like to speak to Mr Charles Bottleneck or to his wife.

ACKNOWLEDGEMENTS AND

REFERENCES

ACKNOWLEDGEMENTS

The author wishes to thank the following for their invaluable assistance while reading for the degree of Doctor of Philosophy.

Professor T. E. Rozzi, Head of the School of Electrical Engineering at the University of Bath, for providing the facilities to carry out this research.

The Royal Signals and Radar Establishment, Malvern, England for providing financial support for 2 years of this research and the University of Bath for providing 6 months financial support.

Dr. E. Whipp for the supervision of this work and the speech research group, formerly of the University of Bath, in particular Brigadier R. A. King, Mr. P. S. Cooper and Mr R. D. Hughes for their invaluable assistance.

To my wife, to whom this thesis is dedicated, for her encouragement and support over the years.

Stephen Longshaw.

References

[1] J. L. Flanagan : Speech Analysis Synthesis and Perception. 2nd. Ed., Springer-Verlag, New York, 1972.

[2] J. N. Holmes : Speech Synthesis. Mills and Boon, London, 1972.

[3] P. Ladefoged : Elements of Acoustic Phonetics. Oliver and Boyd, London, 1962.

[4] J. D. Markel and A. H. Gray : Linear Prediction of Speech. Springer-verlag, New York, 1976.

[5] I. B. Crandall : Sounds of Speech. Bell Syst. Tech. J., 4, No. 4, pp 586-623, Oct. 1925.

[6] T. H. Crowley, G. G. Harris, S. E. Miller, J. R. Pierce and J. P. Runyan : Modern Communications. Columbia University Press, 1962.

[7] C. E. Shannon : A Mathematical Theory of Communications. Bell Syst. Tech. J., 27, No. 3, pp 379-423, JuLY 1948.

[8] H. Dudley : Remaking Speech. J. Acoust. Soc. Am., 11, pp 169-177, 1939.

[9] M. R. Schroeder : Vocoders - Analysis and Synthesis of Speech. Proc. IEEE, 54, pp 720-734, May 1966.

[10] B. Gold and C. M. Rader : The Channel Vocoder. IEEE Trans. Audio Electroacoust., AU-15, pp 148-161, Dec. 1967.

[11] J. N. Holmes : The JSRU Channel Vocoder. Proc. IEE, 127, Pt. F, pp 53-60, Feb. 1980.

[12] J. N. Holmes, Private communication. Joint Speech Research Unit, Cheltenham.

[13] K. W. Cattermole : Principles of Pulse Code Modulation. Iliffe

Books, London, 1973.

[14] J. A. Betts : Signal Processing, Modulation and Noise. ELBS, Bristol, 1975.

[15] N. S. Jayant : Adaptive Quantization with a One-Word Memory. Bell Syst. Tech. J., 52, No. 7, pp 1119-1144, Sept. 1973.

[16] B. Smith : Instantaneous Companding of Quantized Signals. Bell Syst. Tech. J., 36, No. 3, pp 653-709, May 1957.

[17] R. A. McDonald : Signal-to-Noise and Idle Channel Performance of Differential Pulse Code Modulation Systems - Particular Application to Voice Signals. Bell Syst. Tech. J., 45, No. 7, pp 1123-1151, Sept. 1966.

[18] M. D. Paez and T. H. Glisson : Minimum Mean Square Error Quantization in Speech, PCM and DPCM Systems. IEEE Trans. Commun., COMM-20, No. 2, pp 225-230, April 1972.

[19] J. B. O'Neal and R. W. Stroh : Differential PCM for Speech and Data Signals. IEEE Trans. Commun., COMM-20, No. 5, pp 900-912, Oct. 1972.

[20] P. Cummiskey, N. S. Jayant and J. L. Flanagan : Adaptive Quantization in Differential PCM Coding of Speech. Bell Syst. Tech. J., 52., No. 7, pp 1105-1118, Sept. 1973.

[21] B. S. Atal and M. R. Schroeder : Adaptive Predictive Coding of Speech Signals. Bell Syst. Tech. J., 49, No. 5, pp 1973-1986, Oct. 1970.

[22] B. S. Atal : Predictive Coding of Speech at Low Bit Rates. IEEE Trans. Commun., COMM-30, No. 4, pp 600-614, April 1982.

[23] F. de Jager : Deltamodulation, A Method of PCM Transmission Using a 1-Unit Code. Philips Research Report, 7, pp 442-466,

Dec. 1952.

[24] J. E. Abate : Linear and Adaptive Delta Modulation. Proc. IEEE, 55, No. 3, pp 298-308, March 1967.

[25] R. Steele : Delta Modulation Systems. Pen Tech Press, London, 1975.

[26] C. K. Un and D. T. Magill : The Residual-Excited Linear Prediction Vocoder with Transmission Rate Below 9.6 kb/s. IEEE Trans. Commun., COMM-23, No. 12, pp 1466-1474, Dec 1975.

[27] R. E. Crochiere : On the Design of Sub-Band Coders for Low-Bit Rate Speech Communication. Bell Syst. Tech. J., 56, No. 5, pp 747-770, May-June 1977.

[28] R. E. Crochiere and M. R. Sambur : A Variable-Band Coding Scheme for Speech Encoding at 4.8 kb/s. Bell Syst. Tech. J., 56, No. 5, pp 771-779, May-June 1977.

[29] R. Zelinski and P. Noll : Adaptive Transform Coding of Speech Signals. IEEE Trans. Acoust. Speech, Signal Proc., ASSP-25, No. 4, pp 299-309, Oct. 1979.

[30] R. A. King and W. Gosling : Time Encoded Speech. Electronics Letters, 14, No. 15, 20 July 1978.

[31] W. Gosling and R. A. King : Time Encoded Speech - An Approach to Digital Voice Transmission. Telephony, April 14, 1980.

[32] J. C. R. Licklider and I. Pollack : Effects of Differentiation, Integration and Infinite Peak Clipping upon the Intelligibility of Speech. J. Acoust. Soc. Am., Am-20, No. 1, pp 42-51, Jan 1948.

[33] J. C. R. Licklider, D. Bindra and I. Pollack : Intelligibility of Rectangular Speech Waves. Am. J. of Psychology, 51, No. 1,

pp 1-20, Jan. 1949.

[34] L. F. Turner, E. Frangoulis and A. Alcaim : Some considerations relating to the performance of variable-information-rate-source to constant-transmission-rate schemes of data compression. IEE J. Comput. & Digital Tech., 2, No. 3, pp 134-141, June 1979.

[35] D. C. Mason and D. M. Balston : Relationship between system delay and transmision rate in time-encoded speech. Electron. Lett., 16, No. 4, pp 128-130, Feb. 1980.

[36] L. F. Turner, E. Frangoulis and A. Alcaim : Further results on relationship between system delay and transmision rate in time-encoded speech - type systems. Electron. Lett., 16, No. 25, pp 947-948, Dec. 1980.

[37] R. A. King and J. Holbeche : Impact of Entropic Coding on the time delay associated with Time Encoded Speech (TES) systems. Electron. Lett., 17, No. 12, pp 394-396, June 1981.

[38] J. N. Holmes : A Survey of Methods of Digitally Encoding Speech Signals. IERE Int. Conf. Digital Proc. of Signals in Communications Loughborough, April 1981.

[39] M. M. Z. Al-Doubooni : Speech Encoding For Low Data Rate Transmission. Ph.D Thesis, University of Bath, England. 1981.

[40] Miproc 16-bit High Speed Microprocessor Handbook. Plessey Microsystems, Towcester, Northants, England. 1978

[41] I. F. Blake (Ed.) : Algebraic Coding Theory. Benchmark papers in Electrical Eng. and Computer Science Dowden, Hutchinson and Ross, Inc. 1973.

[42] E. R. Berlekamp : Algebraic Coding Theory. Mc Graw-hill, New York, USA. 1968.

[43] S. Lin and D. J. Costello : Error Control Coding. Prentice Hall Inc., New Jersey, USA 1983.

[44] E. N. Gilbert : Synchronization of Binary Messages. IEEE Trans. Inform. Theory, IT-6, No. 4, pp 470-477, Sept 1960.

[45] P. G. Neumann : Efficient Error Limiting Variable Length Codes. IEEE Trans. Inform. Theory, IT-8, No. 4, pp 292-304, July 1962.

[46] P. G. Neumann : On a Class of Efficient Error Limiting Variable Length Codes. IEEE Trans. Inform. Theory, IT-8, No. 5, pp 260-260, Sept 1962.

[47] W. H. Kautz : Fibonacci Codes For Synchronization Control. IEEE Trans. Inform. Theory, IT-11, No. 2, pp 284-292, April 1965.

[48] R. H. Barker : Group Synchronization of Binary Digital Systems. Communication Theory, W. Jackson, Ed., Academic Press, 1953.

[49] J. J. Stiffler : Theory of Synchronous Communications. Prentice Hall, New Jersey, USA 1971.

[50] E. N. Gilbert and E. F. Moore : Variable Length Binary Encodings. Bell Syst. Tech. J., 38, No. 4, pp 933-967, July 1959.

[51] D. A. Huffman : A Method for the Construction of minimum Redundancy Codes. Proc. IRE, 40, pp 1098-1101, 1952.

[52] P. S. Cooper and S. Longshaw : Proposed Formats for the Serial Transmission of TES data. Internal Report, University of Bath, England. Sept. 1981.

[53] M. V. Mathews : Extremal Coding for Speech Transmission. IRE Trans., IT-5, pp 129-136, 1959.

[54] J. C. R. Licklider : Effects of Amplitude Distortion upon the Intelligibility of Speech. J. Acoust. Soc. Am., Am-18, No. 2,

pp 429-434, Oct. 1946.

[55] J. C. R. Licklider : The Intelligibility of Amplitude-Dichotomised, Time Quantized Speech Waves. J. Acoust. Soc. Am., Am-22, No. 6,pp 820-823, Nov. 1950.

[56] W. A. Ainsworth : Relative Intelligibility of Different Transforms of Clipped Speech. J. Acoust. Soc. Am., Am-41, No. 5, pp 1272-1276, Sept. 1969.

[57] V. N. Sobolev and V. N. Telepnev : Simple Methods of Clipped Speech Regeneration. Telecommunications, 23, No. 3, pp 37-44, 1969.

[58] V. J. Phillips and L. D. Thomas : A Feed-Forward Speech Level Controller for Speech Channel Signals. Voice Microsystems Ltd, England. Report No. 14-12-81, Dec. 1981.

[59] L. R. Rabiner and B. Gold : Theory and Application of Digital Signal Processing. Prentice-Hall, 1975.

[60] A. V. Oppenheim and R. W. Schafer : Digital Signal Processing. Prentice-Hall, 1975.

[61] J. W. Tukey : Exploratory Data Analysis. Addison-Wesley, 1971.

[62] B. Justusson : Noise Reduction by Median Filtering. Proc. 4th Int. Joint Conf. on Pattern Recognition. Japan. pp 502-504, Nov. 1978.

[63] L. R. Rabiner, M. R. Sambur and C. E. Schmidt : Applications of a Non Linear Smoothing Algorithm to Speech Processing. IEEE Trans. Acoust. Speech Sig. Proc., ASSP-23, No. 6, pp 552-557, Dec. 1975.

[64] N. S. Jayant : Average and Median-Based Smoothing Techniques For Improving Digital Speech Quality In The Presence of Trans-

mission Errors. IEEE Trans Comm., COM-24, No. 9, Sept. 1976, pp 1043-1045.

[65] T. S. Huang, G. J. Yang and G. Y. Tang : A Fast Two Dimensional Median Filtering Algorithm. IEEE Trans. Acoust. Speech Sig. Proc., ASSP-27, No. 1, pp 13-17, Dec. 1979.

[66] B. R. Frieden : A New Restoring Algorithm For The Preferential Enhancement Of Edge Gradients. J. Opt. Soc. Am., 66, No. 3, pp 280-283, March 1976.

[67] N. C. Gallegher and G. L. Wise : A Theoretical Analysis Of The Properties Of Median Filters. IEEE Trans. Acoust. Speech Sig. Proc., ASSP-29, No. 6, pp 1136-1141, Dec. 1981.

[68] J. W. Tukey : Non Linear (Non Superposable) Methods For Smoothing Data. Congress Record, EASCON, p 673, 1974.

[69] J. C. R. Licklider : The intelligibility of Amplitude-Dichotomized, Time-quantized Speech Waves. J. Acoust. Soc. Am., 22, pp 820-823, 1950.

[70] J. L. Flanagan : A Difference Limen For Vowel Formant Frequencies. J. Acoust. Soc. Am., Am-27, pp 613-617, 1955.

[71] J. L. Flanagan : Difference Limen For The Intensity Of A Vowel Sound. J. Acoust. Soc. Am., Am-27, pp 1223-1225, 1955.

[72] R. A. King, Private communications, Royal Military College of Science (Cranfield).

[73] C. M. Kortman : Redundancy Reduction - A Practical Method of Data Compression. Proc. IEEE, 55, No. 3, pp 253-263, March 1967.

[74] C. A. Andrews, J. M. Davies and G. R. Schwarz : Adaptive Data Compression. Proc. IEEE, 55, No. 3, pp 267-277, March

1967.

[75] L. Ehrman : Analysis of Some Redundancy Removal Bandwidth Compression Techniques. Proc. IEEE, 55, No. 3, pp 278-287, March 1967.

[76] L. D. Davisson : The Theoretical Analysis Of Data Compression Systems. Proc. IEEE., 56, No. 2, pp 176-186, Feb. 1968.

[77] L. D. Davisson and R. M. Gray (Edited by) : Data Compression. Benckmark Papers in Electrical Engineering and Computer Science., 14, Dowden, Hutchinson and Ross, Inc.

[78] G. Benelli, V. Cappellini and F. Lotti : Data Compression Techniques and Applications. The Radio and Electrical Engineer, 50, No. 1/2, pp 29-53, Jan./Feb. 1980.

[79] A. Singh : Hybrid Time Encoded Speech. Ph.D Thesis, Bath University, England. 1982.

[80] G. A. Miller and J. C. R. Licklider : The Intelligibility of Interrupted Speech. J. Acoust. Soc. Am., 22, 2, pp167-173, 1950.

[81] H. P. Kramer and M. V. Mathews : A Linear Coding For Transmitting a Set of Correlated Signals. IRE Trans. on Information Theory, IT-2, pp 41-46, Sept. 1956.

[82] W. R . Crowther and C. M. Rader : Efficient Coding of Vococder Channel Signals Using Linear Transformations. Proc. IEEE (letters), 54, pp 1594-1595, Nov. 1966.

[83] W. K. Pratt, J. Kane and H. C. Andrews : Hadamard Transform Image Coding. Proc. IEEE, 57, No. 1, pp 58-68, Jan. 1969.

[84] S. J. Campanella and G. S. Robinson : A Comparison of Walsh and Fourier Transformations For Applications To Speech. Proc.

1971 Symp. Walsh Functions, Washington D.C, pp 199-205.

[85] S. J. Campanella and G. S. Robinson : A Comparison of Orthogonal Transformations For Digital Speech Processing. IEEE Trans Comm., COM-19, No. 6, pp 1045-1050, Dec. 1971.

[86] F. Y. Y. Shum, A. R. Elliott and W. O. Brown : Speech Processing with Walsh-Hadamard Transforms. IEEE Trans. Audio and Electroacoustics, AU-21, No. 3, pp 174-178, June 1973.

[87] N. Ahmed and K. R. Rao : Data Compression Using Orthogonal Transforms. Proc. 1974 Symp. Walsh Functions, Washington D.C, pp 199-205.

[88] H. Gethoffer : Speech Processing With Walsh Functions. Proc. 1971 Symp. Walsh Functions, Washington D.C, pp 163-168.

[89] N. Ahmed, H. Schreiber and P. Lopresti : On Notation and Definition of Terms Related To a Class of Complete Orthogonal Functions. IEEE Trans. Electromagnetic Compatability, EMC-15, No. 2, pp 75-80, May 1973.

[90] J. L. Shanks : Computation of The Fast Walsh-Fourier Transform. IEEE Trans. Computers, C-18, No. 5, pp 457-459, May 1969.

[91] W. M. Walmsley : Walsh Functions, Transforms and Their Applications. Electronic Eng., pp 63-68, June 1974.

[92] N. Ahmed, A. L. Abdussattar and K. R. Rao : BIFORE or Hadamard Transform. IEEE Trans. Audio and Electroacoustics, AU-19, No. 3, pp 225-235, Sept. 1971.

[93] R. A. King and J. Holbeche, private communications, University of Bath.

[94] R. M. Krauss and P. D. Bricker : Effects of Transmission Delay and Access Delay on the Efficiency of Verbal Communications.

J. Acoust. Soc. Am., Am-41, 2, pp286-292, Feb. 1967.

[95] E. T. Klemmer : Subjective Evaluation of Transmission Delay in Telephone Conversations. B. S. T. J., 46, 6, ppl141-1147, July-August 1967.

[96] J. L. Flanagan, M. R. Schroeder, B. S. Atal, R. E. Crochiere, N. S. Jayant and J. M. Tribolet : Speech Coding. IEEE Trans. Comm, COMM-27, 4, pp710-737, April 1979.

[97] A. Seneviratne, private communication, University of Bath, 1982.

[98] R. G. Gallager : Information Theory and Reliable Communications. John Wiley, 1968.

[99] J. Mills : Good Programming in Assembly Language. IEE Proc., 127, Pt. E, No. 6, pp 241-248, Nov. 1980.

[100] J. L. Walsh : A Closed Set of Orthogonal Functions. Am. J. Mathematics, 45, pp 5-24, 1923.

[101] H. Harmuth : Transmission of Information by Orthogonal Functions. 2nd. Ed., Springer-Verlag, Berlin. 1972.

[102] H. Rademacher : Einige Satze Von Allegemeinen Orthogonalfunktionen. Math. Annalen., 87, pp 122-138, 1922.

[103] R. E. Paley : A Remarkable Set of Orthogonal Functions. Proc. London Math. Soc., 34, No. 2, pp 241-279, 1932.

[104] J. W. Manz : A Sequency Ordered Fast Walsh Transform. IEEE Trans. Audio and Electroacoustics, AU-20, No. 3, pp 204-205, Aug. 1972.

[105] H. Y. L. Mar and C. L. Sheng : Fast Hadamard Transform Using The H Diagram. IEEE Trans. Computers, C-22, No. 10, pp 957-960, Oct. 1973.

[106] K. G. Beachamp : Walsh Functions and Their Applications.

Academic Press, London, 1975.

[107] IEEE Recommended Practice For Speech Quality Measurements. IEEE Trans Audio and Electroacoustics, AU-17, No. 3, pp 227-246, Sept. 1969.

[108] M. E. Hawley (Ed.) : Speech Intelligibility and Speaker Recognition. Benchmark Papars in Acoustics. Dowden, Hutchinson and Ross, Inc. 1977.

[109] W. D. Voiers : The Present State of Digital Vocoding Techniques: A Diagnostic Evaluation. IEEE Trans. Audio and Electroacoustics, AU-16, No.2, pp 275-277, June 1968.

[110] D. Y. Wong and J. D. Markel : An Intelligibility Evaluation of Several Linear Prediction Vocoder Modifications. IEEE Trans. Acoust. Speech Sig. Proc., ASSP-26, No. 5, pp 424-435, Oct. 1978.

[111] W. D. Voiers : Diagnostic Evaluation of Speech Intelligibility Benchmark Papars in Acoustics. Dowden, Hutchinson and Ross, Inc. 1977.